4156 10/18/16

not on test
1st test

testing

what is the difference between unit testing & system testing?

unit tests test code

does the code do what it should at the class & method levels?

system tests test full system

does the system fulfill all requirements? user stories / use cases AND any non-functional requirements

need to do both

acceptance tests = customer's variant of system tests

does the system fulfill our requirements & correctly in our environment

4156 10/18/16

tests can often be automated

invoked by other code, scripts, simulation of a human user, etc.

build process should automatically re-run every developer test after every commit or nightly
⇒ continuous integration

might also include customer tests
⇒ continuous delivery

black box vs. grey box vs. white box

white box- look at code for coverage

grey box- look at intermediate products not visible to end users

e.g. log entries
network I/O

we focus today on black box
others covered later on

4156 10/18/16

black box = focuses on inputs + outputs

functionality - does the unit or system do what it supposed to?

are all user inputs validated?
reject inappropriate inputs
sanitize all user inputs
useful error messages

are all outputs valid?
not just functionally correct
but are they in right format,
not including anything that
shold not leave the system
through that channel
(e.g. sensitive data)

state transitions
is there any way for user to
skip or jump state transitions?
e.g., url hacking to get to
a different state in UI,
back button, browser history
+ cache
(need to check + reject)

4156 10/18/16

broad concept of what is a bug

- sw doesn't do something req. says it shald do
- sw does something that req. says it shouldn't do
- sw does something that req. doesn't mention
- sw doesn't do something that req. doesn't mention but shold
- sw is difficult to undestand, hard to use, slow, etc.

4156 10/18/16

Consider simple calculator

which are bugs in calculator impl. vs. bugs in req.?

- press + key, nothing happens or get wrong answer
- after some period of time or some # of uses, calculator stops responding
- calculator displays all 0's when battery is weak
- calculator does add, subtract, multiply & divide plus undocumented square root when 2 of the buttons are pressed simultaneously
- buttons too small, = key in odd place, display cannot be read in bright light

4156 10/18/16

black box tests written from req., not from code

if we don't know what to test or how to test it, that may be a bug in req. (req. can have bugs too!)

good req. like good code are testable

how do we know when we have a "good" set of tests?

- find all "inputs" to the system, as implied by req.

- not all input comes from human user

- info requested from DB, file, device, network is also an input

- responses from library, system calls, external API calls are also inputs

4156 10/18/16

is there at least one test case that checks each input? (input point, not value)

same for outputs (output point, not value)

for each input point, there might be "infinitely" many possible values

- cannot try all of them (exhaustive testing)
- need to try more than one
- how to decide which to try?

"equivalence partitions" (aka equivalence classes)

- set of values where we expect sw to behave same way (not same result)

for calc, might expect all positive integers (up to max) to behave similarly

but zero or negative might behave differently

4156 10/18/16

positive, zero, & negative intgers are valid equivalence classes for most calculator

also floating point - to some degree of precision

but alphabetic characters are not valid input

set of equivalence partitions should include valid partitions, often more than one, and also invalid partitions, again often more than one

not all invalid ~~inputs~~ input should be treated as "equivalent"

consider dates, with month represented as integer 1 to 12

0, 13, -1, string invalid

but how sw react to invalid may differ - error message, raise exception, crash, hang

4/15 6 10/18/16

consider a calendar program, which does something different based on season

it assumes 12, 1, 2 Winter
3, 4, 5 Spring
6, 7, 8 summer
9, 10, 11 fall

4 different valid equivalence classes

but what if end-user thinks
1, 2, 3 winter
4, 5, 6 Spring
7, 8, 9 summer
10, 11, 12 fall

Categories of equivalence classes

range - below, within, above

size or length - within limits, too big, too small

sets - member or non-member

4156 10/18/16

Container data structures
w/ set of contents &
min/max # of contents

in or not in container
empty or full container
is same element allowed in
container more than once?
must contents be in some order?

special cases for certain data structures

tree - root, interior node, leaf
tree with exactly one node
empty tree
full tree
balanced vs. unbalanced?

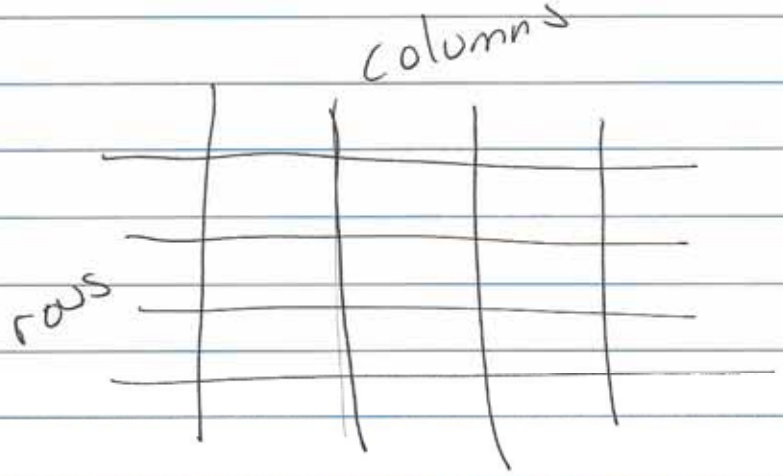
number - negative, zero, positive
maxint, minint
# of decimal points
& / of range of exponent

string - printing vs. non-printing
special characters meaningful
to application
length

int vs. string overflow

4156 10/18/16

table



with or without missing or repeated values
sorted or not ~~sop~~ sorted by a "key" column

file – exists or not
readable, writeable, executable, none (permissions)
correct format or not
min + max size

relationships among multiple inputs must be consistent, check what happens when not
- date of birth & age

need to check for every input point & combination of dependent input points

ideally also check output points (backtrace when input came from)

4156 Continue here

10/18/16

boundary analysis

have you ever heard of an off by one error, a fencepost error, a corner or edge case?

these are all problems that may be missed by equivalence partitions but caught by boundary analysis

instead of (clearly) inside a partition or outside a partition, they're at boundary of partition

a little bit off

obviously, look for literally off by one

min	min-1	min+1
max	max-1	max+1

often caused when code should be < but is <=
or vice versa
(and > vs. >=)

4156 10/18/16

also consider how data is represented in computer

$2^N - 1$, 2^N, $2^N + 1$, 0

default, null, empty, blank, none

may have to get creative about how the off by one concept applies to your inputs

consider how devious or clueless users can be as well, not providing off by one inputs

one character too few or too many
switching lower + upper case (when this matters)
omitting an expected input in a form with multiple slots including space / tab characters where not permitted

4156 10/18/16

wow, that's a lot of test cases!

yes it is

good code often has 3x as many lines of test code as application code

good code also has independent test cases - can be run in any order with same results

test cases, or group test cases, need setup & teardown that makes no assumptions about previous test cases & cleans up after themselves