Thursday 9/18/14 4156

purpose of testing is to find bugs

what is a bug?

1. sw doesn't do something that requirements says it should do
2. sw does something that req says it shouldn't do
3. sw does something that req doesn't mention
4. sw doesn't do something that req doesn't mention but should
5. sw is difficult to understand, hard to use, slow, etc. (catch-all)

Thursday 9/18/14 4156

consider simple calculator

#1 press + key, nothing happens or get wrong answer

#2 after some period of use, stops responding

#3 calculator displays all 0's when battery gets weak

#4. does addition, subtraction, multiplication, division & undocumented square root

#5 buttons too small = key in odd place display cannot be read under bright lights [or dim lights]

which of these are bugs in the calculator vs. bugs in the requirements?

Thursday 9/18/14 4156

TDD makes it easier to find bugs in requirements –

because developer isn't sure what tests to write

or which tests should pass vs. which should fail

assume we are confident in the requirements, & have defined a few test cases

how do we know if this is a "good" set of test cases?

find all the "inputs" to the system, as implied by the use cases

- is there at least one test case for each different "input"?

- not all "input" comes from the user

might do same with all "outputs"

Thursday 9/18/14 4156

consider each input point, not each input value

for each opportunity to provide input, there may be an infinite (or very large) number of possible input values

- cannot try all of them
- but, in most cases, should try more than one
- how to decide which values?

equivalence partitions (or equivalences classes) are sets of inputs values where we expect equivalent results

- valid vs. invalid inputs
- may be multiple invalid partitions

consider months represented as ints

wrong type, e.g., string
below range, e.g., -1 or 0
above range, e.g., 13 or 12345

Thursday 9/18/14 4156

- may be multiple valid partitions

need to consider the functionality defined by (or implied by) the use case

ex. if the system is supposed to do something different based on season

then 12, 1, 2 winter
3, 4, 5 spring
6, 7, 8 summer
9, 10, 11 fall

four different valid equivalence classes & need to test with at least one input value from each

hmm, what if customer assumes
1, 2, 3 winter
4, 5, 6 spring
7, 8, 9 summer
10, 11, 12 fall

Thursday 9/18/14 4156

categories of possible equivalence partitions to consider

for any kinds of input values that can be ordered, there may be range issues
below, within, above

for any kinds of input values that have size or length, then more range issues
below, within, above limits

set - member or non-member

number - negative, zero, positive
number of decimal points
or range of exponent

string - printing vs. non-printing characters
special characters

table - with a without "missing" values
with a without "repeating" values
sorted or not sorted by "key"

Thursday 9/18/14 4156

file - exists or not
readable or not
writeable or not
correct format or not

all these consider a single input point, but there may be required relationships among input points

combinations of input values consistent or not

ex. date of birth & age

equivalence classes can be covered during testing if any one value from each class tested (or combinations when considering related inputs)

but often errors occur at <u>boundaries</u> of equivalence classes

known as fencepost, edge, corner cases
"off by one" error

Thursday 9/18/14 4156

what constitutes a boundary & where to look for them?

min / max	shortest / longest
first / last	soonest / latest
start / finish	highest / lowest
over / under	nearest to / furthest from
empty / full	

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

also consider how data is represented in computer

if data size is N bits consider 2^N-1, 2^N, 2^N+1, 0

ascii character table is not contiguous

0-9	48-57
A-Z	65-90
a-z	97-122

default, empty, blank, null, zero, none, etc.

Thursday 9/18/14 4156

if time permits, divide into groups of ~4

devise set of equivalence partitions for last year's pre-assignment