individual assignment

4156 Thursday 11/7/13 page 1

what is a software bug?

1. the sw doesn't do something that the product spec says it should do
2. the sw does something that the product spec says it shouldn't do
3. the sw does something that the product spec doesn't mention
4. the sw doesn't do something that the product spec doesn't mention but should
5. the sw is difficult to understand, hard to use, or slow (or does something different that the user ~~expects~~ thinks is not right)

consider simple calculator

#1 press + hey, nothing happens
get wrong answer

#2 pound on keys & calc stops responding

exploratory testing ?

#3 does addition, subtraction, multiplication & division AND undocumented square root
#4 the calculator does weird things when the battery gets weak
#5 buttons too small
placement of = key hard to use
display difficult to read under bright lights
catch all

now that we've defined "bug" some other terms

verification - confirm that SW meets it spec
validation - confirm that SW meets the user's requirements
quality - meets customers needs
reliability - stable & dependable
testing - find bugs
quality assurance - Create & enforce standards & methods to improve the SW process & prevent bugs from occurring (or being deployed)

static + dynamic testing

static – examine, don't execute
dynamic – execution

black box static testing
specification + design review

white box static testing
code review

no – cover next Thursday

pretend to be the customer
what are your customer's needs?

~~Is~~ are any corporate terminology or conventions relevant?
– are they adhered to?

what are the industry requirements?
e.g., medical, pharmaceutical, financial

any government standards?

Standard GUI style & guidelines
for Windows, Mac, etc.
Security standards?

Specification attributes checklist?
- complete
- accurate
- precise, unambiguous & clear
- consistent
- relevant
- feasible
- code-free
- testable

move to next Thu

look for problem words
- always, every, all, none, never
- certainly, therefore, clearly, obviously, evidently
- some, sometimes, often, usually, ordinarily, customarily, most, mostly
- etc., and so forth, and so on, such as
- handled, processed, rejected, shipped, eliminated
- if... then... (but misses else)

dynamic black box testing
(+ white box = coverage)
test to pass } get simplest cases
test to fail } working 1st, then
try to break it

how do we select the actual data to
use for test to fail?

- equivalence partitioning
methodically reduce huge (infinite)
set of possible test cases to a
small, but effective, set

ideas to find ~~class of~~ classes
of test cases that are equivalent
in sense of revealing same bug
(or likely to)

consider calculator example
we checked 1+1, 1+2, 1+3, 1+4
- do we need to check 1+5, 1+6?
- how about 1 + 999999.... ?

consider a window filename
it can contain any characters except
\ / : * ? " < > |
1 to 255 characters

equivalence classes for
valid characters
invalid characters
valid lengths
lengths too short
lengths too long

but equivalence classes not enough
since might choose arbitrary
example from "middle"

need to also consider boundaries
since many coding errors made
at boundaries
"off by one" error
> when should be >=
array start at 0 not 1

what constitutes a boundary & where to look for them?

look for certain types

numeric	position / location
character	speed
string	size
	date
	quantity

look at characteristics of type

1st / last	min / max
start / finish	over / under
empty / full	shortest / longest
slowest / fastest	soonest / latest
largest / smallest	highest / lowest
nearest-to / furthest-from	

then what actually to test?

min, min-1, min+1

max, max-1, max+1

→ also consider sub-boundary or internal conditions

how data is represented in computer

if data size is 16 bits,
consider $2^{16}-1$, 2^{16}, $2^{16}+1$
generally 2^N-1, 2^N, 2^N+1
also 0
if signed, then $+2^{N-1}-1$, 2^{N-1}, $2^{N-1}+1$

another example: ascii character table
not nice contiguous list

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

need to consider just below & just above alphanumeric ranges for many applications

also consider when relevant
default, empty, blank, null, zero, none

and thrown total garbage testing
if has #s, great letters
not printable chars, etc