CS 130 SOFTWARE ENGINEERING

HOARE LOGIC:
A FORMAL INSPECTION TECHNIQUE

Professor Miryung Kim
UCLA Computer Science
Based on Materials from Miryung Kim

All copyrights are reserved by Professor Kim and UCLA.

AGENDA

- Hoare Triples: A Formal Inspection Technique
- Software inspection methods
- Code reviews & Pair programming

RECAP: MODEL BASED TESTING

- White box structural testing with focus on control structure
- Black box testing --- leverage models (UML activity chart, finite state machine, grammar)
- Systematic + exhaustive ➔ randomization / probabilistic behavior modeling
- Mutation testing. You create a set of mutants (i.e., Program versions with injected faults), measure a mutant kill ratio.

SOFTWARE INSPECTION OVERVIEW

COLLABORATIVE CONSTRUCTION

- pair programming
- formal inspections
- informal technical reviews

PAIR PROGRAMMING BENEFITS (1)

- Pair programming can achieve code quality similar to formal inspections (Shull et al 2002).
- The cost of pair programming is 10-25% higher than the cost of solo development, but the reduction in development time is 45%.

PAIR PROGRAMMING BENEFITS (2)

- IBM found that each hour of inspection prevented about 100 hours of related work
- Raytheon reduced the cost of defect correction from 40% to 20% via emphasis on software inspections
- HP reported that its inspection program saved an estimated $21.5 million per year

FORMAL INSPECTIONS

- An inspection is a specific kind of view, shown to be effective in detecting defects
- Developed by M. Fagan in IBM

FORMAL INSPECTION (1)

- Checklist focus the reviewers' attention on areas that have been problems in the past
- The inspection focuses on defect detection not correction
- Reviewers prepare for the inspection meeting beforehand and arrive with a list of problems they've discovered

FORMAL INSPECTION (2)

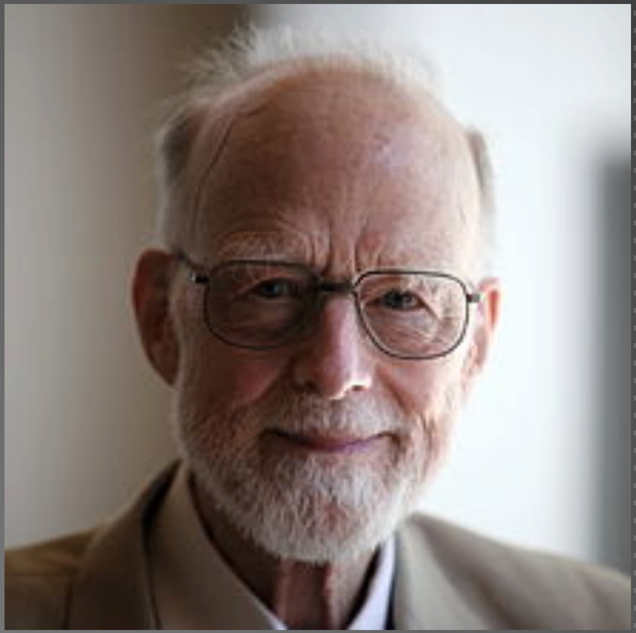
- Distinct roles are assigned to all participants
- The moderator of the inspection isn't the author of the work product under inspection.
- The moderator has received specific training in moderating inspections

ROLES DURING AN INSPECTION

- Moderator is responsible for keeping the inspection moving.
- Author is a person who wrote the design or code.
- Reviewer has a direct interest in the code but not the author. Her/his role is to find defects. They usually find defects during preparation, as the design is discussed at the meeting.
- Scribe records errors that are detected and the assignments of action items during the inspection meeting.
- Management - is not a good idea to include in the meeting.

HOARE LOGIC

HOARE LOGIC



- Hoare-style Program Verifications
- Sir Tony Hoare
- Known for Hoare logic
- Communicating Sequential Processes
- was at Oxford University, now moved to Microsoft Research
- Some of the slides are borrowed from K. Rustan Leino@ MSR (HP Lab/ ESC Java)

PEER REVIEW SCENARIO

```
public char[] foo(Object x, int z)
  if (x != null) {
     n = x.f;
   } else {
     n = z-1;
     z++;
   }
  a = new char[n];
 return a;
}
```

Suppose Alice wrote `foo`. **Which arguments need to be passed to foo** so that it returns a non null value without throwing any `NullPointerException` and `ArrayOutOfBoundException`?

EXAMPLE

which pre-condition should hold here?

```
if (x != null) {
   n = x.f;
} else {
   n = z-1;
   z++;
}
a = new char[n];
```

true

WHY DO WE NEED TO KNOW ABOUT HOARE LOGIC?

- Suppose that your colleague wrote the code.
- Which arguments /inputs do you need to have, not to crash her code?
- Computing weakest preconditions can find **subtle bugs** and **corner cases** during peer **code reviews.**

MY PERSONAL VIEW…

- Writing "pre-conditions" and "post-conditions" is important for code comprehension, collaboration, code review, and bug finding.
- Yet, people do not write pre-/post conditions, or write a very shallow pre-post conditions.
- Therefore, I am teaching Hoare Logic so that you can write pre- and post-conditions effectively.

STATE PREDICATES

- A predicate is a Boolean function on the program state
- Examples:
- $x == 8$
- $x < y$
- $m \leqq n \Rightarrow (\forall j \mid 0 \leqq j < a.length \cdot a[j] \neq NaN)$
- true
- false

HOARE TRIPLES

- For any predicates P and Q and any program S,

postcondition

$$\{P\}\ S\ \{Q\}$$

precondition

says that if S is started in (a state satisfying) P, then it terminates in Q

EXAMPLES

- $\{true\}\ x := 12\ \{x = 12\}$
- $\{x < 40\}\ x := 12\ \{10 \leq x\}$
- $\{x < 40\}\ x := x+1\ \{x \leq 40\}$
- $\{m \leq n\}\ j := (m+n)/2\ \{m \leq j \leq n\}$

PRECISE TRIPLES

- If $\{P\}\ S\ \{Q\}$ and $\{P\}\ S\ \{R\}$, then does $\{P\}\ S\ \{Q \wedge R\}$ hold??

PRECISE TRIPLES

- If $\{P\}\ S\ \{Q\}$ and $\{P\}\ S\ \{R\}$, then does

$$\{P\}\ S\ \{Q \wedge R\}$$

hold?

PRECISE TRIPLES

- If $\{P\}\ S\ \{Q\}$ and $\{P\}\ S\ \{R\}$, then does

  $$\{P\}\ S\ \{Q \wedge R\}$$

  hold? yes

- The most precise Q such that

  $$\{P\}\ S\ \{Q\}$$

  is called the strongest postcondition of S with respect to P.

WEAKEST PRECONDITIONS

- If $\{P\}\ S\ \{R\}$ and $\{Q\}\ S\ \{R\}$, then

  $$\{P \lor Q\}\ S\ \{R\}$$

  holds.

- The most general P such that

  $$\{P\}\ S\ \{R\}$$

  is called the weakest precondition of S with respect to R,

  written wp(S, R)

TRIPLES AND WP

$\{P\}\ S\ \{Q\}$

if and only if

$P \Rightarrow wp(S, Q)$

PROGRAM SEMANTICS
—SKIP

- no-op
- $wp(\mathbf{skip}, R) \equiv R$
- $wp(\mathbf{skip}, x^n + y^n = z^n)$
  $\equiv x^n + y^n = z^n$

PROGRAM SEMANTICS
—ASSERT

- if P holds, do nothing, or else terminate
- wp(assert P, R) $\equiv$ P $\wedge$ R
- wp(assert x < 10, 0 $\leqq$ x)
- == x<10 AND x>=0
- == 0<=x<10
- wp(assert x = y*y, 0 $\leqq$ x)
  == x=y^2 AND x>=0
- == x=y^2
- wp(assert false, x $\leqq$ 10)
- ==false AND x<=10
- ==false (precondition false means that there exists no input value that will execute the program statement and ends up with a post condition x<=10

PROGRAM SEMANTICS —ASSERT

- if P holds, do nothing, else don't terminate
- wp(assert P, R) $\equiv$ $P \wedge R$

- wp(assert x < 10, 0 $\leqq$ x)
- $\equiv$ 0=<x<10
- wp(assert x = y*y, 0 $\leqq$ x)

  $\equiv$ x=y^2 AND x>=0

$\equiv$ x=y^2

wp(assert false, x $\leqq$ 10)

=false AND x<=10

=false

PROGRAM SEMANTICS
—ASSIGNMENT

- evaluate E and change value of w to E
- wp(w := E, R) ≡ R[w := E]

replace w by E in R

- wp(x := x + 1, x ≦ 10)
  == {x+1<=10}
- == {x<=9}
- wp(x := 15, x ≦ 10)
  == {15 <=10}
- == false (there is no input that will go through x:=15 and end up with postcondition x<=10)
- wp(y := x + 3*y, x ≦ 10)
  == {x<=10}
- wp(x,y := y,x (short for swapping values), x < y)
  == {y<x}

PROGRAM SEMANTICS —ASSIGNMENT

- evaluate E and change value of w to E
- $wp(w := E, R) \equiv R[w := E]$

replace w by E in R

- $wp(x := x + 1, x \leqq 10)$
  $\equiv x<=10\ [x:=x+1]$
- $\equiv x+1<=10$
- $\equiv x<=9$
- $wp(x := 15, x \leqq 10)$
  $\equiv x<=10\ [x:=15]$
  $\equiv 15<=10$
  $\equiv$ false
- $wp(y := x + 3*y, x \leqq 10)$
  $\equiv x<=10\ [y:=x+3y]$
  $\equiv x<=10$
- $wp(x,y := y,x, x < y)$
  $\equiv x<y[x:=y,y=x]$
- $\equiv y<x$

PROGRAM COMPOSITIONS

- If $\{P\}\ S\ \{Q\}$ and $\{Q\}\ T\ \{R\}$, then $\{P\}\ S\,;T\ \{R\}$ (; is a sequential execution)
- If $\{P \wedge B\}\ S\ \{R\}$ and $\{P \wedge \neg B\}\ T\ \{R\}$, then $\{P\}$ if B then S else T end $\{R\}$

PROGRAM SEMANTICS —SEQUENTIAL COMPOSITION

- wp(S;T, R) ≡ wp(S, wp(T, R))

- wp(x := x+1 ; assert x $\leqq$ y, x<0)
  == wp (x:=x+1, wp (assert x<=y, {x<0}))

- == wp (x:=x+1, {x<=y AND x<0}) == {x+1<=y AND
  {x+1<=y and x<-1}

- wp(y := y+1 ; x := x + 3*y, y $\leqq$ 10 $\wedge$ 3 $\leqq$ x)
  == wp (y:=y+1, wp (x:=x+3y, {y<=10 AND 3<=x}))
  == wp (y:=y+1, {y<=10 AND 3<=x+3y})

- == {y+1<=10 AND 3<=x+3(y+1)}

- == {y+1<=10 AND 3<=x+3(y+1)}

- == {y<=9 AND 3<=x+3y+3} == {y<=9 AND 0<=x+3y}

PROGRAM SEMANTICS
—SEQUENTIAL COMPOSITION

- $wp(S;T, R) \equiv wp(S, wp(T, R))$
- $wp(x := x+1 ; \text{assert } x \leqq y, 0 < x)$
  $\equiv \quad wp(x := x+1, wp(\text{assert } x \leqq y, 0 < x))$
  $\equiv \quad wp(x := x+1, 0 < x \leqq y)$
  $\equiv \quad 0 < x+1 \leqq y$
  $\equiv \quad 0 \leqq x < y$
- $wp(y := y+1 ; x := x + 3*y, y \leqq 10 \wedge 3 \leqq x)$
  $\equiv \quad wp(y := y+1, wp(x := x+3*y, y \leqq 10 \wedge 3 \leqq x))$
  $\equiv \quad wp(y := y+1, y \leqq 10 \wedge 3 \leqq x+3*y)$
  $\equiv \quad y+1 \leqq 10 \wedge 3 \leqq x+3*(y+1)$
  $\equiv \quad y < 10 \wedge 3 \leqq x + 3*y + 3$
  $\equiv \quad y < 10 \wedge 0 \leqq x + 3*y$

PROGRAM SEMANTICS —CONDITIONAL COMPOSITION

- wp(if B then S else T end, R) $\equiv$
  (B $\Rightarrow$ wp(S, R)) $\wedge$ ($\neg$B $\Rightarrow$ wp(T, R)) $\equiv$
  (B $\wedge$ wp(S, R)) $\vee$ ($\neg$B $\wedge$ wp(T, R))

- wp(if x < y then z := y else z := x end, 0 $\leqq$ z)
  == { x<y AND wp (z:=y, {0<=z}) OR (x>=y AND wp (z:=x, {0<=z})}
  == { x<y AND 0<=y} OR (x>=y AND wp (z:=x, {0<=z})}

- == { x<y AND 0<=y} OR {x>=y AND 0<=x}

- wp(if x≠10 then x := x+1 else x := x + 2 end, x $\leqq$ 10)
  == {x!=10 AND wp (x:=x+1, {x<=10}} OR (x==10 AND wp (x:=x+2, x<=10}

- == {x!=10 AND x+1<=10} OR {x==10 AND x+2<=10}

- == {x<=9} OR {x==10 AND x<=8}

- == {x<=9} OR FALSE == {x<=9} // there is something wrong with the code, and its not possible to go through else side and produce a desired post condition.

PROGRAM SEMANTICS —CONDITIONAL COMPOSITION

- wp(if B then S else T end, R) $\equiv$
  $(B \Rightarrow wp(S, R)) \wedge (\neg B \Rightarrow wp(T, R)) \equiv$
  $(B \wedge wp(S, R)) \vee (\neg B \wedge wp(T, R))$

- wp(if x < y then z := y else z := x end, $0 \leqq z$)

$$
\begin{aligned}
&\equiv \quad (x < y \wedge wp(z := y, 0 \leqq z)) \vee \\
&\qquad\;\; (\neg(x < y) \wedge wp(z := x, 0 \leqq z)) \\
&\equiv \quad (x < y \wedge 0 \leqq y) \vee (y \leqq x \wedge 0 \leqq x) \\
&\equiv \quad 0 \leqq y \vee 0 \leqq x
\end{aligned}
$$

- wp(if $x \neq 10$ then x := x+1 else x := x + 2 end, $x \leqq 10$)

$$
\begin{aligned}
&\equiv \quad (x \neq 10 \wedge wp(x := x+1, x \leqq 10)) \vee \\
&\qquad\;\; (\neg(x \neq 10) \wedge wp(x := x+2, x \leqq 10)) \\
&\equiv \quad (x \neq 10 \wedge x+1 \leqq 10) \vee (x = 10 \wedge x+2 \leqq 10) \\
&\equiv \quad (x \neq 10 \wedge x < 10) \vee \text{false} \\
&\equiv \quad x < 10
\end{aligned}
$$

RECAP (1)

- We learned about how to reason about the semantics of assertions, assignment, sequential statements, conditional statements.
- These techniques can help you to find subtle errors during peer code reviews.

RECAP (2)

- Though experienced developers may not use the term, "Hoare Logic or Weakest Preconditions", that's how they should read code to find bugs during peer code review.
- Next lecture, we will study how to reason about the behavior of loops & loop invariants.

QUESTIONS?