CS 130 SOFTWARE ENGINEERING

HOARE LOGIC:

A FORMAL INSPECTION TECHNIQUE

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

AGENDA

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

SOFTWARE INSPECTION OVERVIEW

COST OF FIXING BUGS

- The longer a defect remains in the system, the more expensive it becomes to remove

COLLABORATIVE CONSTRUCTION

- pair programming
- formal inspections
- informal technical reviews

PAIR PROGRAMMING

- 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

- 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

PAIR PROGRAMMING BENEFITS

- Each hour spent on inspections avoided an average of 33 maintenance hours
- 55% of one-line maintenance changes were in error before code reviews were introduced.

KEYS TO SUCCESS WITH PAIR PROGRAMMING

- Support pair programming with coding standards
- Don't let pair programming turn into watching
- Don't force pair programming of the easy stuff
- Rotate pairs and work assignments regularly
- Encourage pairs to match each other's pace

KEYS TO SUCCESS WITH PAIR PROGRAMMING

- Make sure both partners can see the monitor
- Don't force people who don't like each other to pair
- Avoid pairing all newbies
- Assign a team leader

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)

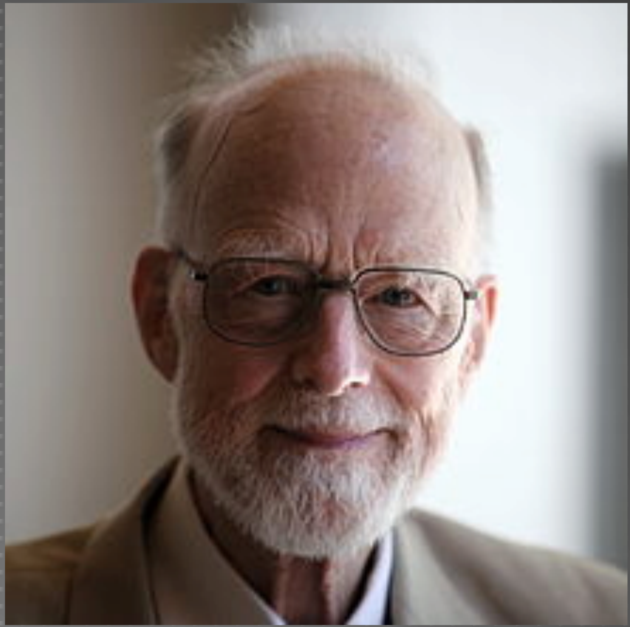
- Distinct roles are assigned to all participants
- The moderator of the inspection isn't the author of the work product under inspection.
- The moderator has received specific training in moderating inspections

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.**

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

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

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

$$wp(\textbf{skip}, R) \quad \equiv \quad R$$

$$wp(\textbf{skip}, x^n + y^n = z^n)$$
$$\equiv \quad x^n + y^n = z^n$$

PROGRAM SEMANTICS
—ASSERT

if P holds, do nothing, else don't terminate

wp(assert P, R) $\equiv$ $P \land R$

wp(assert x < 10, 0 $\leqq$ x)

$\equiv$ x<10 AND 0<=x

$\equiv$ 0<=x<10

wp(assert x = y*y, 0 $\leqq$ x)

$\equiv$ $x = y*y \land 0 \leqq x$

$\equiv$ $x = y*y$

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

$\equiv$ false

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+1<=10

$\equiv$ x<=9

wp(x := 15, x $\leqq$ 10)

$\equiv$ 15<=10

$\equiv$ false

wp(y := x + 3*y, x $\leqq$ 10)

$\equiv$ x<=10

wp(x,y := y,x, x < y)

$\equiv$ 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

$$wp(x := x + 1, x \leqq 10)$$
$$\equiv \quad x+1 \leqq 10$$
$$\equiv \quad x < 10$$

$$wp(x := 15, x \leqq 10)$$
$$\equiv \quad 15 \leqq 10$$
$$\equiv \quad \text{false}$$

$$wp(y := x + 3*y, x \leqq 10)$$
$$\equiv \quad x \leqq 10$$

$$wp(x,y := y,x, x < y)$$
$$\equiv \quad y < x$$

PROGRAM COMPOSITIONS

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

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

$$wp(S;T, R) \quad \equiv \quad wp(S, wp(T, R))$$

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

$\equiv$

wp $\equiv$

$\equiv$

$\equiv$

$wp(y := y+1 \;;\; x := x + 3*y,\ y \leqq 10 \wedge 3 \leqq x)$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

PROGRAM SEMANTICS —SEQUENTIAL COMPOSITION

$$wp(S;T, R) \quad \equiv \quad wp(S, wp(T, R))$$

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

$\equiv \quad wp(x := x+1,\ wp(\text{assert } x \leq y,\ 0 < x))$

$\equiv \quad wp(x := x+1,\ 0 < x \leq y)$

$\equiv \quad 0 < x+1 \leq y$

$\equiv \quad 0 \leq x < y$

$wp(y := y+1 \,;\, x := x + 3*y,\ y \leq 10 \wedge 3 \leq x)$

$\equiv \quad wp(y := y+1,\ wp(x := x+3*y,\ y \leq 10 \wedge 3 \leq x))$

$\equiv \quad wp(y := y+1,\ y \leq 10 \wedge 3 \leq x+3*y)$

$\equiv \quad y+1 \leq 10 \wedge 3 \leq x+3*(y+1)$

$\equiv \quad y < 10 \wedge 3 \leq x + 3*y + 3$

$\equiv \quad y < 10 \wedge 0 \leq x + 3*y$

PROGRAM SEMANTICS —CONDITIONAL COMPOSITION

$$wp(\text{if } B \text{ then } S \text{ else } T \text{ end}, R) \equiv$$
$$(B \Rightarrow wp(S, R)) \land (\neg B \Rightarrow wp(T, R)) \equiv$$
$$(B \land wp(S, R)) \lor (\neg B \land wp(T, R))$$

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

$\equiv$

$\equiv$

$\equiv$

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

$\equiv$

$\equiv$

$\equiv$

$\equiv$

PROGRAM SEMANTICS —CONDITIONAL COMPOSITION

$$
\begin{aligned}
&wp(\text{if } B \text{ then } S \text{ else } T \text{ end}, R) \equiv \\
&(B \Rightarrow wp(S, R)) \wedge (\neg B \Rightarrow wp(T, R)) \equiv \\
&(B \wedge wp(S, R)) \vee (\neg B \wedge wp(T, R))
\end{aligned}
$$

$$
\begin{aligned}
&wp(\text{if } x < y \text{ then } z := y \text{ else } z := x \text{ end}, 0 \leq z) \\
\equiv\quad & (x < y \wedge wp(z := y, 0 \leq z)) \vee \\
& (\neg(x < y) \wedge wp(z := x, 0 \leq z)) \\
\equiv\quad & (x < y \wedge 0 \leq y) \vee (y \leq x \wedge 0 \leq x) \\
\equiv\quad & 0 \leq y \vee 0 \leq x
\end{aligned}
$$

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

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 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?