

FORMALLY CORRECT

Dr. Christelle Scharff

Assistant Professor of Computer Science

Pace University, New York, NY

Presentation at Institute of Technology of
Cambodia

June 13rd 2002

OUTLINE

- Software Engineering Process
- What does 'CORRECT' mean?
- Why formal methods?
- Where are formal methods used?
- Formal methods
- Algebraic Specifications
- JAVA and formal methods
- Conclusion

SOFTWARE ENGINEERING PROCESS

- Activities in all software processes:
 - **Specification** – what the software should do and its development constraints.
 - **Development** – production of the software.
 - **Validation/Verification** – Checking/Verifying that the software is what the customer wants.
 - **Evolution** – changing the software in response to changing demands.

EXTREME PROGRAMMING

- Software engineering process model
 - Extreme Programming: delivery of small increment of functionalities, constant code improvement, involvement of the user in the development team and pair-programming.

CORRECT?

- No syntax error (compiler, interpreter).
- No semantics error.
- Correct:
 - The program has all the properties we want it to ascertain.
 - The program does what we want it to do.

TOWARD CORRECT PROGRAMS

- A *correct program* is not *just more* reliable
it is reliable.
- A *correct program* does not *rarely* goes wrong
it cannot go wrong.
- A *correct program* does not *almost* solve a problem
it solves a problem.
- “*The correct program should be the philosopher stone for the programmer, the pole star of his efforts.*” Andrew Cumming.

TESTING VERSUS VERIFYING

- Testing is widely used.
- Roughly 60% are development costs and 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Testing = Considering different (hopefully **all**) inputs and checking that the program returns the right outputs.
 - Validation of the software – No guarantee.
- Verifying = **Proving** that the software does what we want it to do with respect to a (formal) specification.
 - Formal methods.
- Verification and testing can be coupled.

WHY TO USE FORMAL METHODS?

Some software horror stories.

- The 1998 shooting down of the Airbus 320 by the US Vincennes was attributed to the cryptic and misleading output displayed by the tracking software.
- A China Airlines Airbus Industries A300 crashes on April 26, 1994 killing 264 people. Recommendations include software modifications.
- The Ariane 5 satellite launcher malfunction was caused by a faulty software exception routine resulting from a bad 64-bit floating point to 16-bit integer conversion.

- An iraqi Scud missile hit Dhahran barracks leaving 28 dead and 98 wounded. The incoming missile was detected by the Patriot defenses, whose clock had drifted 36 seconds during the 4-days continuous siege, the error increasing with elapsed time since the system was turned on. This software flaw prevented real-time tracking. *ACM SIGSOFT Software Engineering Notes*
- Eighteen errors were detected during the 10-day flight of Appolo 14. *G.J Myers, Software Reliability: Principles and Practice.*
- An error in a single FORTRAN statement resulted in the loss of the first American probe to Venus. *G.J Myers, Software Reliability: Principles and Practice.*

- Five nuclear reactors were shut down temporary because a program testing their resistance to earthquakes used an arithmetic sum of variables instead of the square root of the sum of the squares if the variables. *H. Lin, scientific American*
- INTEL processor bugs galore.
- The nine-hour breakdown of AT&T's long distance telephone network in January 1990, caused by an untested code patch, dramatized the vulnerability of complex computer systems everywhere. *Ghost in the Machine, Time Magazine, January 1990*

INDUSTRIAL USE OF FORMAL METHODS

–IN FRANCE–

- Aviation: Dassault Aviation (Esterel, Coq)
- Nuclear: Schneider Electric SA (Lustre)
- Transportation: Steria (B) ...

INDUSTRIAL USE OF FORMAL METHODS

– IN THE UNITED STATES–

- Databases: HP Medical Instruments, real-time database for storing patient monitoring information (1991)
- Nuclear: Argonne National Laboratories work on reactor safety
- Security: Security policy model for the NATO Air command and control system
- Telephone: Various features of the AT&T telephone switching were verified using Esterel (1996)

FORMAL METHODS

- A method is **formal** if it has sound mathematical basis.
- Framework to specify, develop and verify software in a systematic rather than ad hoc manner.
- Formal software development:
 - **Specification** – using a specification language
 - **Proof of properties** on the specification (theorem provers and deduction systems)
 - Consistency
 - Completeness
 - Correctness
 - **Implementation** – (property-preserving) transformation of the specification

WHAT IS A SPECIFICATION LANGUAGE?

- It provides:
 - a **syntax**
 - a **semantics**
 - **rules** defining which objects satisfy each specification
- Examples:
 - Algebraic specification
 - Programming languages
 - Programs can be viewed as formal objects that we can manipulate.
 - ...

PROPERTIES OF SPECIFICATIONS

- **Unambiguous** (one meaning)
 - Natural language is informal and ambiguous.
- A specification must be **consistent**.
 - No contradiction.
- A specification should be **complete**.
 - All scenarios are specified.
- An implementation is **correct** with respect to a specification if it implements the specification:
 - all the specification and only it
 - all properties of the specification must be preserved

PROVING PROPERTIES

- Using a **theorem prover**.
- A theorem prover provides rules (how to deduce new information from current information) and strategies on the rules (how to apply the rules).

FORMAL METHODS EXAMPLES

- **Model-oriented formal methods** (semantics) – Construction of a model of the software by enumeration
 - Examples: Z, VDM, Petri nets, Unity. . .
- **Property-oriented formal methods** (syntactic) – State a property to be verified by the software and use an inference system to prove it
 - Examples: B, LARCH, PVS, algebraic specification. . .
- **Executable formal methods** – Logic and functional programming languages and Programming languages in general.
- **Visual formal methods**
 - Examples: Petri nets, statecharts...

EXAMPLE OF AN ALGEBRAIC SPECIFICATION

Type nat

Constructors:

zero : -> nat

suc : nat -> nat

Operations:

plus : nat -> nat

infe : nat * nat -> bool

Definitions of the operations:

x, y: variables

plus(zero,x) = x (i.e. 0+x=x)

plus(suc(x),y) = suc(plus(x,y))

(i.e. (x+1)+y=(x+y)+1)

infe(zero,zero) = true

infe(zero,suc(x)) = true

infe(suc(x),suc(y)) = infe(x,y)

infe(suc(x),zero) = false

We prove using the replacement of equal by equal that :

$$\text{infe}(\text{suc}(\text{suc}(\text{zero})), \text{suc}(\text{zero})) = \text{false}$$

FORMAL METHODS AND JAVA

- Programming in JAVA with **Preconditions** and **Postconditions**.

- `// PRECONDITION`
`PROGRAM CODE`
`// POSTCONDITION`

- A precondition is a condition that is true before the code executes.

- A postcondition is a condition that is true if the precondition is true and the code executes completely.

- Examples:

```
// Precondition:  $x < 0$ 
```

```
y = x * -2;
```

```
//Postcondition:  $x < 0$  AND  $y > 0$  AND  $y = -2x$ 
```

```
// Precondition:  $x \geq 0$ 
```

```
x = x % 5;
```

```
// Postcondition:  $x' \geq 0$  AND  $x' \leq 4$ 
```

FORMAL RULES FOR JAVA

- **Assignment, Conditional, Loop rules.**
- Consider the following given JAVA code for an assignment:

```
//P  
x = E;  
//Q
```

- where x is a variable, E is an expression, P is a precondition and Q is a postcondition.
- **Assignment rule:** We derive P by replacing all occurrences of x in Q with E .
- The assignment rule is a means of reasoning backward from a goal back to the required starting conditions.

THE CASE OF LOOPS IN JAVA

- Considering a loop, a **loop invariant** I is true in the following four locations:

```
// Invariant
while (C)
{
// Invariant
loop body
// Invariant
}
// Invariant
```

- Example:

```
sum = 0;
J = 0;
while (J < N)
{
J = J + 1;
sum = sum + J;
}
Invariant = { sum = 1+2+...+J }
```

WHILE rule

- Consider the following given JAVA code for a loop:

```
// P
S1;
while (B)
{
S2;
}
// Q
```

– where B is a Boolean expression, $S1$ and $S2$ contain assignment statements, P is a precondition, Q is a postcondition and I is the loop invariant.

– The **While** rule is the following:

1. $//P \ S1 \ //I$
2. $//I \ S2 \ //I$
3. $I \ \text{AND} \ \text{NOT} \ B \rightarrow Q$
4. The loop terminates.

CONCLUSION

- Software Quality
- Gap between the formal world and the real world
- Lots of challenges in formal methods
 - Hardware verification is more advanced than software verification.
 - Demonstrate that existing techniques can be applied to real examples
 - Educate people to use formal methods
- My research:
 - Theorem provers and deduction systems dealing with equalities

– Application in formal verification of software

For example, the following formula is of the kind one encounters in verifying programs involving array indexing: $(i = j \text{ and } k = l \text{ and } a[i] = b[k] \text{ and } j = a[j] \text{ and } m = b[l]) \Rightarrow a[m] = b[k]$.

– Extreme programming

- Thank you.