Formal Specification can help with code update, refactoring, etc.

The programs proofs have to be maintained throughout code evolutions:

it ensures non-regression of the code.

FIXING CODING ISSUES, LIKE CORRECTING BUFFER OVERFLOWS, ARE REFACTORINGS

Automatic code generation is the solution of the maintainability / performance problem. (It doesn't matter if the generated code is completely unreadable, because you never have to look at it.)

1. Debugging & fault localization is still in the 70s

2. Formal methods has an important role to play

3. The shortage of common benchmarks & infrastructure makes research harder than it has to be.

4 The soup was better than the wrap.

Wireless

VCC or ~~Freeman~~

Access Code: ~~icsme2014~~

~~icsmereg2014~~

CORRECT ACCESS CODE:

icsme2014

1. HAVE CODE ANALYSIS TOOLS FOR THE DISCOVERY OF SECURITY VULNERABILITIES EVOLVED SUFFICIENTLY IN TERMS OF FUNCTIONALITY, FLEXIBILITY AND EASE OF USE TO BECOME MANDATORY PARTS OF THE SOFTWARE CERTIFICATION PROCESS ?

2. WOULD THE WIDESPREAD ADOPTION OF STRONGLY-TYPED LANGUAGES BE A BETTER INVESTMENT IN TERMS OF SOFTWARE SECURITY THAN THE INTEGRATION OF CODE ANALYSIS TOOLS IN THE SOFTWARE DEVELOPMENT PROCESS ?

3. WHY AREN'T THERE MORE SOFTWARE SECURITY PRESENTATIONS AT SCAM ??