Opportunities and Obstacles to Using Static Analysis for the Development of Safety-Critical Software
FAA: use of RTCA DO-178B is an “acceptable means of compliance” with airborne software requirements
  - Advisory Circular 20-115B

Problem: regulatory constraints make software development better, but more expensive and slower
  - Especially verification – review/analysis, test

Goal: Faster, cheaper software verification without making it worse

Solution: use automation tools
Verification Tools

- Test case/vector generators & qualified test coverage tools significantly automate test

- Code review/analysis: enforce coding standard; early elimination of errors
  - Buffer/stack overflow, uninitialized/unused variables, etc.
  - At least one additional engineer examines every line of code
  - Follow-up reviews to verify corrections

- Manual reviews are effective, but time consuming & expensive
  - Also, tedious & susceptible to human error

- Desire: automate source code reviews
• FAA/DER: Provide confidence at least equivalent to the process being automated
  – Demonstrate determinism: same output for same input operating in same environment
  – Demonstrate compliance with operational requirements

• What are static analyzer operational requirements?
  – Error detection accuracy?
  – Detection of what kinds of errors?
  – What tests verify requirements?
  – Who runs the tests & who guarantees independence?

• Task: determine effectiveness of static analyzers & their readiness for qualification
Evaluation of Static Analyzers

• Evaluated 20 static analyzers – open source & COTS
  – Detection accuracy (true positive, true negative, false positive, false negative)
  – Remediation advice (error description; elimination advice)
  – False positive suppression
  – Rule extension
  – User interface
  – IDE integration

• Error classes based on MITRE’s Common Weakness Enumeration
  – Over/underflow/range, type/cast, arithmetic, resource management, pointer, looping errors, etc.

• Analysis accuracy tests based on NIST’s SAMATE Reference Dataset
• None addressed all tested error classes

• Few had high detection accuracy
  
  – All had high false negative rate against 1 or more classes
    unsigned int i = 10;
    while (i >= 0)
    {
      i = i-1;
    }
  – Many had high false positive rate (high usage cost)

• Few had clear remediation advice

• Few scaled well from components to large systems

• Some were difficult to use, difficult to integrate
Conclusions

- Accuracy, ease of integration & use make some cost effective
  - Detect some errors faster & better than manual reviews
  - Some reduce downstream costs far beyond usage cost (e.g., false positive analysis)

- None can completely replace manual review
  - Manual review still finds errors static analyzers currently do not

- Tool qualification is an unresolved obstacle
  - Testable requirements & criteria are emerging
  - Business case for vendor provided qual is unclear – e.g., market size, frequency of re-qual
• Without mitigating action, static analysis could degrade the current verification process
  – Developers get false confidence – aware of tool strengths, unaware of weaknesses
  – Defer to tool output & put little effort into manual review
  – Increased errors found in downstream activities

• Some static analyzers might replace manual review within specific error classes
  – Define error classes & operational requirements
  – Set detection accuracy thresholds
  – Define qualification methods & tests against error class
Industry Actions

- **Source Code Security Analysis Tool Functional Specification**
  - NIST-led government, academe, industry group’s functional spec for static analyzers

- **Common Weakness Enumeration (CWE)**
  - MITRE standard definitions of ~300 software error classes

- **SAMATE Reference Dataset (SRD)**
  - NIST/MITRE standard set of ~2000 test cases for static analyzers
  - C, C++, Java; false negative, false positive

- **NIST plan for certifying independent labs for software product qualification testing**
  - Similar to hardware product certification since 70’s
• **Support NIST plan**
  - Approval of source code analysis spec
  - Additional errors & tests for CWE & SRD inclusion
  - Detailed CWE to SRD mapping
  - Additional capability from tool industry
  - Domain-specific CWE & SRD profiles (e.g., avionics) to avoid irrelevant testing

• **Resolve qualification issues with vendors & FAA/DER**
  - Detailed operational requirements
  - Detection accuracy criteria
  - Requirements-based qual tests & results analysis
  - Qual package approval