Technical Guidelines Development Committee
April 20, 2005 Plenary Meeting

Coding standards

What they are

- Mostly, requirements on the *form* (not *function*) of source code
- Some requirements affecting software integrity, implemented as coding practices
  - Error checking
  - Exception handling
  - Prohibit practices that are known risk factors for latent software faults and unverifiable code

Why they are

- Started in 1990 VSS, expanded in 2002
- TGDC Resolution #29-05, “Ensuring Correctness of Software Code” (part 2)
- Enhance readability, maintainability, integrity, verifiability, trustworthiness of software
- Generally accepted software engineering principles

Where they are

- Vol. I Ch. 4 and Vol. II Ch. 5 of 2002 VSS
- Change-tracked revisions in Appendix A and B of draft VVSG2
- To be merged into VVSG 2

The 2002 spec

- Mixture of mandatory and optional
- Vendors may substitute “published, reviewed, and industry-accepted coding conventions”
- Incorporated conventions have suffered from rapid obsolescence and limited applicability
- Some mandatory requirements had unintended consequences

Recommended changes

- Expand coding conventions addressing software integrity
  - Make range checking requirements more explicit
  - Require structured exception handling, “formal exception handlers provided by the language” — I.4.2.3.e
- Retain requirements of high import for logic verification (subject to revision)
- Disincorporate the rest; require use of “published, credible” coding conventions

Credible ≈ industry-accepted

- Coding conventions shall be considered credible if at least two different organizations with no ties to the creator of the rules or to the vendor seeking qualification independently decided to adopt them and made active use of them at some time within the three years before qualification was sought.

Issues

- May need to define more precisely what qualifies as coding conventions or modify definition of credible
- Public comment, April 14: “The NASED Technical Committee has previously ruled that assembler code is permitted as long as the code meets all other requirements.” In tabulation code? Need rationale.
- C doesn’t have structured exception handling
- Disincorporate integrity requirements, etc. if “published, credible” replacements are found

Non-issues

- Assembly language in “hardware-related segments” and operating system software
- Grandfathering of stable code — part of general grandfathering strategy (not for NIST to recommend or determine)
- COTS or “slightly modified” COTS — part of COTS strategy, driven by security requirements, T.B.D.

Discussion