Software Assurance Metrics and Tool Evaluation

Paul E. Black

National Institute of Standards and Technology

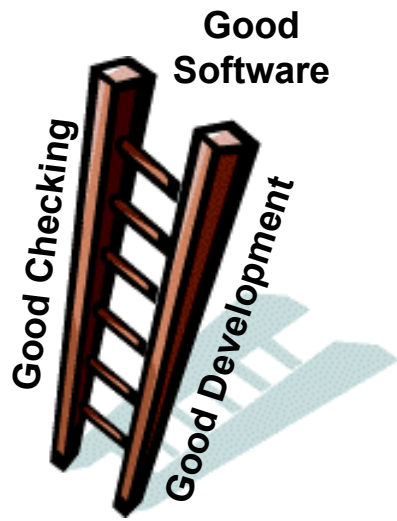
http://www.nist.gov/

paul.black@nist.gov

NIST

What is Software Assurance?

- **Activities that ensures that software processes and products conform to requirements.**
  - **after NASA Software Assurance Guidebook**
- **Two legs of good software**
  - **Good Development**
  - **Good Checking**
    - **Testing (dynamic)**
    - **Analysis (static)**



Why Concentrate on Checking?

- **Vital for software developed outside, i.e., when process is not visible**
- **Applicable to legacy software**
- **Feedback for process improvement**

- **Process experiments are expensive**
- **Many are working on process (SEI, etc.)**

The NIST SAMATE Project

1. **Surveys**
   - Tools
   - Researchers and companies
2. **Workshops & conference sessions**
   - Taxonomy of SA functions and techniques
   - Order of importance (cost/benefit, criticalities, …)
   - Gaps and research agendas
   - Studies to develop metrics
3. **Tool evaluations**
   - Detailed specification
   - Test plans and reference material
   - Reports of tool evaluation

Workshops

- **List SA functions and techniques**
  - **Approach (code vs. spec, static vs. dynamic)**
  - **Software type (distributed, real time, secure)**
  - **Type of fault detected**
- **Which are the most "important"?**
  - **Highest cost/benefit ratio?**
  - **Finds highest priority vulnerabilities?**
  - **Most widely used?**
- **Identify gaps in functions or tools**
- **Plan and initiate studies for metrics**

Purposes of SA Tool Evaluations

- **Precisely document what a tool does (and doesn't) do**

  … in order to …

- **Provide feedback to tool developers**
  - **Simple changes to make**
  - **Directions for future releases**
- **Inform users**
  - **Match the tool to a particular situation**
  - **Understand significance of tool results**

Details of SA Tool Evaluations

- **Develop clear (testable) requirements**
- **Develop a measurement methodology:**
  - **Test cases**
  - **Procedures**
  - **Reference material**
  - **Scripts and auxiliary programs**
  - **Interpretation criteria**
- **Evaluate tools**
- **Publish evaluations**

But, are the Tools Effective?

Do they really find vulnerabilities and catch bugs? In other words, how much assurance does running a tool provide?

Toward Software Metrics

- **Qualitative comparison**

  warmer, colder *buggy, secure*

- **Formally defined quantity**

  temperature *quality? confidence?*

- **Unit and scale**

  degree, kelvin ?

- **Measured value**
- **Derived units**

  Heat energy=smt *Software assurance≈pt*

Society has 3 options:

- **Learn how to make software that *works***
- **Limit size or authority of software**
- **Accept failing software**