Topics

- Traditional Software V&V Approach
- Modified Software V&V Approach for Systems V&V
- Use of Software/Systems Integrity Levels
- V&V of COTS and Reused Software
- V&V Techniques for Systems Analysis
- Cost Estimating for Systems V&V
- Future Direction of IEEE Std 1012 – Systems V&V
Traditional Software V&V Approach

System Segment Specifications

Allocate to CIs

Software V&V

- Assign software integrity levels (SIL) to critical functions
- Apply minimum V&V tasks (IEEE Std 1012)
- Add optional V&V tasks IAW system needs
- Apply appropriate intensity and rigor to high SIL functions

Software Component Item (CI)

Hardware Component Item (CI)

User/Operator

#1 #2 #N = System Requirements

X = Critical System Function

= System Requirements
System/Software Integrity Level (SIL)

- Range of values that represent software complexity, criticality, risk, safety level, security level, desired performance, reliability, or other project-unique characteristics that define the importance of the software to the user.

- SIL determination & assignment
  - Determined in the context of the system
  - Vary depending on intended application/use of system
  - Assigned to software requirements, functions, groups of functions, or software components or subsystems
  - May change as the software evolves
  - Use system SIL if one exists
  - Establishes intensity & rigor of V&V tasks for high integrity SIL software functions
SIL Assignment Example

- Design feature
- Technology insertion

SIL may change or remain unchanged

- Error
Systems V&V Approach

System Segment Specification & CONOPS

System Scenarios

Software Component Item (CI)

Hardware Component Item (CI)

User/Operator

System Scenarios:

- System #1
- System #2
- System #N = System Requirements

X = Critical System Function

Copyright 2005 Northrop Grumman Corporation
Systems V&V Methodology

- Systems V&V is performed within the context of the operating environment and involves:
  - V&V of each key component
    - Software CI
    - Hardware CI
    - Operator instructions/manuals
  - V&V of interactions of CIs
    - Analysis of critical system scenarios across CIs
    - Maximizing system performance, safety or security instead of optimizing individual CI’s performance
Proven Systems V&V Techniques

- System Level Control/Data Flow Analysis
  - Critical function threads
- Scientific Modeling/Simulation
  - Integration across long time periods
- Analysis of System Performance Events
  - Throughput
  - Data transaction profile
  - Robust error handling
- Analysis of Off Nominal Events
  - Event Singularities
  - Out of Sequence Events
  - System Component Deterioration
V&V of COTS & Reused Software

- **What is it?**
  - Software from software libraries
  - Custom software developed for other applications
  - Commercial-off-the-shelf (COTS) software
  - Artifacts from existing software (requirements, design, etc)

- **Software Developed By Reuse Process**
  - Analyze artifacts of the domain engineering
    - Plans, models, architecture
    - Significant analysis of the domain engineering products during all software life cycle processes

- **Software Developed Outside of Reuse Process**
  - Special consideration required during the V&V effort when:
    - Inputs required for V&V not available
    - Reused SW developed
      - For system different in function or application from intended use
      - Meets different user needs than current system
    - Original user needs are unknown
V&V of Reuse Software

Reuse Candidate Software Development

Reuse Engineering Process

Development

Reuse Dev Artifacts

Reuse Source

Reuse Object

V&V

Reusability Analysis

Reuse V&V Artifacts

No Reuse Engineering Process

Development

Reuse Dev Artifacts

Reuse Source

Reuse Object

V&V

Reuse V&V Artifacts

System Software Development

Development

Dev Artifacts

Source

Object

IEEE Std 1012

Table 2

V&V Tasks

V&V

Reuse Analysis

Table D1-D4

V&V Tasks & Table 2 Tasks as appropriate

Dashed line items may not exist
**V&V of Reuse Software: Techniques**

<table>
<thead>
<tr>
<th>Substitution Tasks:</th>
<th>Substitute alternative analysis and test methods in lieu of 1012 requirement V&amp;V tasks to generate objective conclusions about the correctness, completeness, accuracy, and usability of the reused software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique #1:</td>
<td>“Black Box Testing”</td>
</tr>
<tr>
<td>Technique #2:</td>
<td>“Review Developer’s QA”</td>
</tr>
<tr>
<td>Alternative Sources:</td>
<td>Use alternative sources of program data to derive conclusions about the correctness, completeness, accuracy, and usability of the reused software.</td>
</tr>
<tr>
<td>Technique #3:</td>
<td>“Operational History”</td>
</tr>
<tr>
<td>Technique #4:</td>
<td>“Audit Results”</td>
</tr>
<tr>
<td>Technique #5:</td>
<td>“Artifacts”</td>
</tr>
<tr>
<td>Reverse Engineering:</td>
<td>Reverse engineer requirements, design, and code data to generate objective conclusions about the correctness, completeness, accuracy, and usability of the reused software.</td>
</tr>
<tr>
<td>Technique #6:</td>
<td>“Reverse Compilation”</td>
</tr>
<tr>
<td>Independent Prototyping and Comparison:</td>
<td>Develop a model (prototype) of the proposed software or use portions of the prior system. Execute test scenarios on the prototype or prior system and compare test results. Analyze the results to generate objective conclusions about the correctness, completeness, accuracy, and usability of the reused software.</td>
</tr>
<tr>
<td>Technique #7:</td>
<td>“Prototyping”</td>
</tr>
<tr>
<td>Technique #8:</td>
<td>“Prior System Results”</td>
</tr>
</tbody>
</table>
## Technique #1: “Black Box Testing”

<table>
<thead>
<tr>
<th>Pros:</th>
<th>Cons:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Test results reflect actual software</td>
<td>➢ Inability to detect errors if error not observable (e.g., latency errors)</td>
</tr>
<tr>
<td>➢ Limits propagation of errors through interfaces</td>
<td>➢ Limit checks may not cover all scenarios</td>
</tr>
<tr>
<td>➢ Have system and user requirements derived from User’s Manual</td>
<td>➢ Requirements check limited by quality of User’s Manual</td>
</tr>
<tr>
<td>➢ Independent analysis</td>
<td></td>
</tr>
</tbody>
</table>

1. **Black Box Testing and Validation:** Execute a spectrum of test cases and validate the correctness of the output.

2. **User’s Manual Analysis:** Derive system and software requirements from the User’s Manual. Use for validating test results.

3. **Limit Checks in Interfacing Software:** Add limit checks in the interfacing software (logic and data) to ensure no erroneous information is passed across the interface.
Technique #3: “Operational History”

1. **Historical Data Analysis:** Analyze the operational history of the reused software with attention to how the software performed in a system with similar characteristics to the new system being proposed.

2. **User Interviews:** Conduct interviews with operational users. Focus data gathering on how the system performed in scenarios and conditions similar to those expected in the new system being proposed.

**Pros:**
- Real data in an operation environment
- User observations about the performance of the software and its related system
- Software burn-in established and track record of discrepancies recorded
- Independent analysis

**Cons:**
- Different characteristics, technologies, and user interfaces could cause errors not observed in the historical system
- Not all interactions recordable or observable so data completeness and accuracy are limited
- User observations can be subjective, biased, and error-prone
Technique #6: “Reverse Compilation”

1. Reverse Engineer Source Code:
   Reverse compile “pseudo source” code from the program object file. Analyze the pseudo code using normal V&V tasks including all V&V test strategies and methods.

2. Reverse Engineer Requirements:
   Derive the system and software requirements from the user’s manual. Analyze the requirements using the 1012 V&V tasks and test the reused software against these requirements.

<table>
<thead>
<tr>
<th>Pros:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Uses actual code – no hidden or implied data</td>
</tr>
<tr>
<td>➢ Perform all 1012 V&amp;V tasks</td>
</tr>
<tr>
<td>➢ Test data reflects actual performance of the reused software using conditions of the proposed system</td>
</tr>
<tr>
<td>➢ Independent analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cons:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Time consuming to reverse engineer data</td>
</tr>
<tr>
<td>➢ Pseudo code hard to read</td>
</tr>
</tbody>
</table>
Technique #8: “Prototyping”

1. Comparison of Prototype Code: Develop a replication model (prototype) of the function or requirements in a user-friendly development environment. Execute test cases representing the range of system scenarios on both the model and reuse software. Compare the results and analyze the differences to determine whether the reused software is performing as intended.

Pros:
- Useful for small functions and sets of requirements
- Easy to diagnose problems in reuse software
- Ability to run a wide range of system scenarios and compare against benchmark program
- Independent analysis

Cons:
- Cost and time of building the model
- Errors in the model can mask errors in the reuse software (likelihood of two similar errors generated by two independent sources should be small or unlikely)
ANSI/IEEE Std 1012: Software V&V

1012-1986

- Plan Standard
- Critical/Non Critical
- Required Tasks
- Optional Tasks

1012-1998

- ISO/IEC 12207
- Process Standard
- SW Integrity Levels
- Required Tasks/Levels
- Elaboration of Tasks (Table 1)
- Entry./Exit Criteria
- Notions on:
  a) V&V Metrics
  b) Reuse Software
- Definition of IV&V

1012-2004

- Continued Process
- Broad industry application
- State of the Art V&V Practices
  a) Reused SW
  b) Metrics
  c) Process Improvement (CMMI)
- Systems Viewpoint
- Measures of V&V effectiveness and efficiency

1012-20??

- ISO/IEC Adoption
- Broad industry application
- Systems V&V
  a) Systems (Part I)
  b) Software (Part II)
  c) Hardware (Part III)
  d) Operator (Part IV)
- State of the Art Practices (required)
  a) Reuse
  b) Metrics