DIY Software Validation

David Vogel, Ph.D.
Intertech Engineering Associates, Inc.
MD&M Midwest - Chicago - 2008

David Vogel
781-801-1117
dav@inea.com
www.inea.com

Copyright 2008 – Intertech Engineering Associates, Inc.

———

Schedule of Presentations

• Overview of the AAMI TIR
• Introduction & Regulatory Background
• Creating a Corporate Process for Validating Production and Quality System Software
The Technical Information Report (TIR)

Where to Order Your Copy of

AAMI TIR36:2007 - Validation of Software for Regulated Processes

http://www.aami.org/applications/search/details.cfm?webid=P855_D4668

or link from Intertech home page:

www.innea.com

Background on TIR Workgroup

• Backgrounds
• Timeline
• What Made This a Difficult Subject
• Outcome
AAMI TIR36 Contents

- General (Purpose & Intent)
- Regulatory Context
- Software Validation Discussion
- Definitions
- Confidence-Building Activities
- Tools In The Toolbox
- Critical Thinking
- Software Validation Details
- Regulated?
- Process Description
- Lifecycle Validation Activities
- Brief Narrative on Documentation & Prerequisite Processes
- Annex A - The Toolbox
- Annex B - Risk Management
- Annex C - Several Examples

AAMI TIR

“Validation of Software for Regulated Processes”

- Detailed software development and validation lifecycle activities
- Reader may need to scale down for OTS, embedded, etc.
- Includes concepts of “Critical Thinking”
- Describes a “Validation Toolbox”
- Discusses Risk Assessment and Risk Management
- Describes a “Validation Process”
- Examples

Validation Work Streams

Additional Content for Maintenance and Retirement Activities
Key Contributions (opinion)

- Thinking Critically
  - It’s OK to do the right thing
  - Excel vs Spreadsheet – Access vs Queries & Reports
  - Reliance on other elements of process to build confidence
- Thinking Outside the Box
  - Effective Validation Considers the Process
  - Applying Risk Management Methodologies
  - Slightly different from device RM, but effective for focusing validation activity
- There’s not a Single “Right Way”
  - Examples show diversity of approaches
- Many Other Tools & Techniques From Software Validation Literature Applied to Regulated Process Software

Validation and Regulatory Background

Verification & Validation

- You need to understand what the terms mean to understand what you must do, and to understand how you should “think critically” about doing it.
• Verification Activities
  • Builds confidence that your software accurately implements your requirements and designs

• Validation Activities
  • Builds confidence that your software is what your users (stakeholders) actually need, and can use. Note that Verification Activities are a subset … the users need accurately implemented software.

Common Misconceptions

1. Validation = Testing  X
2. Verification = Testing  X
3. V&V = Testing  X

Three Different, But Related Meanings
Why Are Verification and Validation Important for You?

- To help reduce the risk of an unsafe failure of the software... And to satisfy regulatory requirements (compliance).
- NOTE: Compliance is simply a side effect of doing what is in your own best interest **NOT** the other way around.
  - It is important to understand this point to adopt a successful attitude about validation
  - If you feel that you are wasting your time “validating” software, you probably are.
End Section 1

Doing It Yourself

Basic “Red Zone” Activities for Non-Software Engineers Validating Their Own Software

So … You’re Interested in Validating Your Software

• What software should non-professional software or software quality engineers attempt to validate?
  • “Modest” applications
    • Criticality and complexity
  • Examples:
    • Narrow scope, off-the-shelf software
    • Modest custom software
    • Spreadsheets, database reports, macros
    • Software embedded in production tools
The Basic Ingredients – Part I

1. What are you using the software for, who will use it, and how? (Intended Use)
2. What features and functions must the software have to satisfy your needs? (Requirements)
3. Plan what you will do to build and maintain your confidence in the software throughout the life of the software (lifecycle planning & management)

The Basic Ingredients – Part II

4. What could happen if the software does fail, and what will you do to minimize the harmful effects? (Risk Analysis and Management)
5. Manage the versions and variants (configuration management)
6. Testing to build confidence in
   • functionality used,
   • any risk control measures implemented
   • Appropriateness for the intended use, by intended users, in intended environment

One More Thing

• If you are going to do all the right things ...

• DOCUMENT IT !!!
   • So you will remember
   • So your successors will know
   • So your boss recognizes your value
   • …. and so regulatory inspectors can see your good work
Intended Use and Requirements

- Subtle but distinct difference
- Example: Text Editor
  - Intended Use: “To create, edit, and format ASCII text files that will be used as software source files and make files that will be subsequently post-processed and checked for syntax errors, and whose executable versions of the software will be subject to their own validation”

Points to Note

- Our definition of the intended use determines the scope of the validation effort
- In this example, we may rely on the syntax checking of the post-processing syntax checking compiler, or make utility in our verification of the outputs of the editor
- Note that this means we are not generically validating for any and all intended uses.
- Each new type of use must re-examine the validation rationale to determine if it is still applicable ... if not, enhance the existing validation package, or create a new one specific to the new intended use

A Question for You

- Would the text editor example above be validated to edit existing ASCII numeric control (NC) files for a laser cutting machine tool?
- What questions would you ask yourself to determine the answer to this question?
So What are the Requirements?

• For our "modestly" critical and complex software we DON’T mean reverse engineering a complete detailed software requirements specification (SRS).
• We DO mean documenting the functions and features that have to work reliably for the software to meet the needs for the intended use.
• For “ambitiously” critical or complex custom applications a complete requirements specification IS appropriate to decompose user needs into specific requirements for software behavior.

Lifecycle Planning
Basic Validation Ingredient 3

• Define Phases of the Life of the Software (e.g. Requirements, Design, Implement, Test, Maintenance, Retirement … OR ... Procurement, Test, Deployment, Maintenance, Retirement, etc.)
• What Validation activities should you Plan for each phase to build and maintain your confidence in the software.
• How do you choose or define a lifecycle? What question would you ask yourself to help decide?

Risk Analysis and Management
Basic Validation Ingredient 4

• What damage or ultimate harm to users, operators, bystanders could result from improper functioning of software?
• Severity of some (all?) risks may be dependent on intended use
• If severity of risk would not be tolerable should it occur, what will you do to prevent or mitigate the damage or harm so that it would be tolerable?
Configuration Management Plan
Basic Validation Ingredient 5

• How many versions will there be?
• Who approves changes to the software?
• How will you control unapproved changes from being made (esp. embedded production sw)?
• How will you identify version?
• How will you track what version runs where?
• Are there other items SW version must coordinate with? (other SW, HW, processes, etc.) How will you control that?

Test Protocols
Basic Validation Ingredient 6

• When you have nothing left … test.
• Spend your test budget where it will do the most good
  • Most critical functions
  • Most complex
  • Most likely to fail (how do you know?)
  • Least documented or understood
• Why is testing a weak risk control measure?
A Short Story

• Very Simple Production Support Software
• Complicated, time consuming, costly issues.
• Take note of the problems experienced.
• After Story, we’ll talk about some fundamentals, then see how the Story might have had a better series of outcomes.

Verification Example
EXTRUDO-CALC

Calculate an extrusion rate based on x-section area of part.

User Needs – Intended Use

Requirements

1. Operator shall input x-section area
2. Operator shall input material density
3. Rate shall be calculated as A*D
4. Software shall output rate as y.y lb/min.

Design & Implementation

Input “Area: ”, A
Input “Density: ”, D
R=A*D
Output “Rate is “,R,” for area “,A,” and density “,D.

Testing

Test Cases:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>-1</td>
<td>err</td>
</tr>
</tbody>
</table>

Note:
Revision cycle would have been anticipated in req verification.

User Needs – Intended Use

Requirements

1. Operator shall input x-section area
2. Operator shall input material density
3. Rate shall be calculated as A*D
4. Software shall output rate as y.y lb/min.

Design & Implementation

Input “Area: ”, A
Input “Density: ”, D
R=A*D
Output “Rate is “,R,” for area “,A,” and density “,D.

Testing

Test Cases:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>-1</td>
<td>err</td>
</tr>
</tbody>
</table>

What Happened During Verification Testing?

• Requirements development and review (verification) did not anticipate invalid inputs
• Testing of software (verification of implementation of requirements) anticipated error handling (although requirement was missing), and failed.
• The software was fixed. All tests passed. Everything’s OK … or is it?
Problems

- Software was shipped to manufacturing facility in Ireland.
- The next day Ireland phones to complain that the area and extrusion rate are in English units, but their cross-sections are in cm² and the devices are set in metric units (kg/hr). Since they need to calculate a conversion for the input and output, the software does little good for them.

Intended Use Environment

- The software team failed to think about intended use environment, and did not test in the intended use environment.
- Intended Use Environment testing is an example of VALIDATION testing (not verification testing).
- Note that after verification testing, the software worked properly in the American plant … but that was not the only intended use environment!

What Happened Next?

- The software team modified the software to make a metric version and sent it to Ireland. The Ireland plant is happy that their new software tool is now working.
- What else could go wrong?
Success?

• Things go well for the company. Sales increase, but there is pressure to reduce costs.
• An additional manufacturing plant is built in China.
• China is on the metric system, so the software team sends a copy of the Irish software to China.

Maybe Not

• Next day, the Chinese plant calls to inquire what was on the CD they just received. Nobody can read the English language prompts.
• What went wrong?

Intended Users

• The intended user changed this time.
  • Language became a consideration.
The Ultimate Solution?

• The software team decided that they should move to an icon based user interface to avoid any future language issues. (They’re catching on)
• The Chinese software is modified, tested, and shipped to all manufacturing plants.

Not Quite?

• The next day, the American plant reports that they are having problems with the extrusion line.
• The newly modified Chinese icon based version was also metric. The attempt to consolidate and re-use software failed to anticipate the units difference.

Why Did That Happen?

• Configuration Management failure.
  • The software had evolved over a number of years.
  • The development team had experienced turnover.
  • The number of variants and versions of the software had not been documented, tracked or controlled.
  • The software was out of control.
Back to EXRUDO-CALC
Basic Validation Ingredient 1

• Intended Use:
  • "To calculate extrusion rate settings on the Extrudo 1 and Extrudo 2 extruders used to manufacture the XYZ Product. The calculation of rate should be based on cross-sectional diameter of extruded part and density of material in our manufacturing plants worldwide."

• Notes:
  • Scope limited to Extrudo 1 and Extrudo 2 calcs
  • No limit on location i.e. language

EXTRUDO-CALC Requirements
Basic Validation Ingredient 2

• User prompts and inputs shall be icon based (language-less or internationally recognized)
• The units for cross-sectional area input shall be configurable at installation time for metric (cm²) or English (in²) units
• The units for density input shall be …
• The units for the output of extrusion rate shall be …
• Prior to calculation of extrusion rate the Area and Density shall be converted to appropriate units for calculation of the Rate in the configured rate units.
• The extrusion rate shall be calculated as A * D

EXTRUDO-CALC Lifecycle Plan
Basic Validation Ingredient 3

• Define Phases of the Life of the Software (e.g. Requirements, Design, Implement, Test, Maintenance, Retirement … OR … Procurement, Test, Deployment, Maintenance, Retirement, etc.)
• What Validation activities should you Plan for each phase to build and maintain your confidence in the software.

• How do you choose or define a lifecycle? What questions would you ask yourself to help decide?
  Maintenance and Retirement Issues
Risk Analysis and Management
Basic Validation Ingredient 4

• What damage or ultimate harm to users, operators, bystanders could result from improper functioning of software?
• Severity of some (all?) risks may be dependent on intended use
• If severity of risk would not be tolerable should it occur, what will you do to prevent or mitigate the damage or harm so that it would be tolerable?

Risk Analysis and Risk Management for EXTRUDO-CALC

• Damage to Parts that could harm patients
  • Extrudo-1 makes tubing … could impact pt.
    • Control: QC tests on tubing to assure proper tubing wall thickness
  • Extrudo-2 makes foam pillows
    • Patient harm unlikely or minor severity … no control
    • HOWEVER, the foam process runs at much higher pressures and a software (or data entry) error resulting in too much pressure could lead to operator injury
      • Control options: Add a requirement to not allow output of unsafe rates and pressures. Add requirement for alert message in this situation. Add pressure relief hardware to make Extrudo-2 safe even for excessive pressures … etc.

EXTRUDO-CALC Configuration Management Plan
Basic Validation Ingredient 5

• How many versions will there be?
• Who approves changes to the software?
• How will you control unapproved changes from being made (esp. embedded production sw)?
• How will you identify version?
• How will you track what version runs where?
• Are there other items SW version must coordinate with? (other SW, HW, processes, etc.) How will you control that?
**EXTRUDO-CALC - Test Protocols**

**Basic Validation Ingredient 6**

- When you have nothing left … test.
- Spend your test budget where it will do the most good
  - Most critical functions
  - Most complex
  - Most likely to fail (how do you know?)
  - Least documented or understood
- Why is testing a weak risk control measure?

---

**Summary**

- Validation and Testing are NOT synonymous
- Even 3 lines of software can be difficult to control, and can result in bad consequences
- Think about the right thing to do to protect patients, users, and your business from software failure. It generally leads you to the right thing to do from a regulatory perspective.