How the Fundamental Assurance Question pervades Certification

Martin S. Feather
Jet Propulsion Laboratory,
California Institute of Technology

© 2013 California Institute of Technology. Government sponsorship acknowledged. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program. This activity is managed locally at JPL through the Assurance and Technology Program Office.
Certification

Building **confidence** in order to help make a decision, e.g.,

- Buy/not buy
- Trust as is/verify
- Release to I&T/hold for more work
- Launch?

Often guided by prescriptive, process-based methods – mandated practices, e.g.,

- Configuration Management
- Requirements traced to design & test
- Tracking and closure of bugfixes

Confirmed by

- IV&V; software assurance policies & procedures
Fundamental Assurance Question

How much do the activities of the prescriptive standard contribute to the confidence needed for certification?

Need to be able to answer this to know how much assurance is enough!

Instead, **Assurance Cases** organize the assurance argument along these lines:
Assurance Cases


“A structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment”

[“Defence Standard 00-56”. U.K. Ministry of Defence, June 2007]

Often contrasted with “prescriptive” certification standards, e.g.,

Prescriptive (process based) vs. Argument based

Prescriptive, e.g., DO178C:
Level of safety criticality => Lifecycle activities
  e.g., extent of test coverage (Statement, Decision or MC/DC)

Argumentative [?]
Prescriptive (process based) AND Argument based

Both depend on adequate software development practices to have been conducted well, e.g.,

- Configuration Management
- Requirements traced to design & test
- Tracking and closure of bugfixes

Prescriptive standards direct this through mandated practices (IV&V; software assurance policies & procedures)

Assurance cases provide for reporting this within “backing arguments & evidence”

HOW MUCH IS ENOUGH?
Inability to answer “How much is enough?” has led to Underappreciating Assurance

“The purpose of assurance is to discover significant* defects, and thus decrease risk”
* = ones that increase risk

Certainly desirable when it does so, but fails to value assurance when it confirms that things are OK, hence:

• **Dismisses** majority of assurance efforts as valueless
• **Potentially misdirects** the choices of when, where and what assurance practices to perform
Assurance activities & findings w.r.t. Standard Processes

<table>
<thead>
<tr>
<th>Process</th>
<th>Task Planning</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare for software task planning</td>
<td>Perform high-level planning for software</td>
<td>Manage project risk assessment</td>
</tr>
<tr>
<td>Support software development</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Red = finding
Green = assured OK
Grey = no activity in this area

Hierarchy of Standard Processes rendered as rectangles divided into rectangles divided into...
Assurance activities & findings w.r.t Standard Processes

**INSIGHTS ➔ QUESTIONS**

Why are these not being covered?

Hot Spots – so what?

What’s the value of assurance if it results in no findings?

---

<table>
<thead>
<tr>
<th>Assurance</th>
<th>Support</th>
<th>Task Planning</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>需确认</td>
<td>安全和恢复</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>配置管理</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>测试和产品质量保证</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>工程</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>需求和开发管理</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>设计</td>
<td>计划任务</td>
<td>管理</td>
</tr>
<tr>
<td>需确认</td>
<td>产品和工程</td>
<td>计划任务</td>
<td>管理</td>
</tr>
</tbody>
</table>

---

9
The Vicious Cycle of Poor SA

Project does not see value in SA

- Insufficient project funding for SW Assurance
- Little to no added value given to most projects due to lack of SA resources
- SMA Management does not see value in SA & lacks project funding for
- Few SA personnel hired and those not fully supported
- Too few SA personnel cover too many projects
- SA Work is scattered and ineffective

Graphic courtesy of Martha Wetherholt, NASA SW Assurance Technical Fellow
Redrawn from “Figure 9 – Complacency …” in “Archetypes for Organizational Safety,” K. Marais & N. Leveson (2006), Safety science 44(7)
Certification: building confidence is the key

“Valuing Certification of Software Release Readiness,”
D. Port & J. Wilf (2013),
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)

A software release is a decision to deliver code to an organization outside of the development team...

The release decision is primarily based on the understanding of the level of quality the software currently has...

A poor understanding of the quality level increases decision risk...

Certification of release readiness attempts to address this risk by building confidence in the quality level.
Building Confidence through Sampling

A demonstration.
Instead of Bernoulli’s colored pebbles in an urn...

White Bear = YOU WIN

Orange Moose = YOU LOSE

(Beware if you ever think you see an orange moose)
DECISION: Which bag should you draw from?

Remember, Orange Moose = YOU LOSE

Bag A

Bag B

A confident decision is easy with perfect information (but can still be unlucky in the draw)
Decision: Which bag should you draw from if you couldn’t see inside?

Bag X

Bag Y

Without any information, no confidence in deciding either way
Sampling can provide guidance to the decision of which bag to draw from?

Bag X

Bag Y

A random draw from Bag X

Some confidence to choose X

Remember, Orange Moose = YOU LOSE
Which bag should you draw from if you couldn’t see inside?

Bag X

Bag Y

A random pair from Bag X

Full confidence to choose X
Which bag should you draw from if you couldn’t see inside?

Bag X

A random pair from Bag X

Back to no confidence in decision

But if you discard the orange moose, a little confidence to choose X
Which bag should you draw from if you couldn’t see inside?

Bag X

Bag Y

A random triple from Bag X

Full confidence to choose Y
And this is like assurance because...

Bag = software development artifact

White bear = a portion that is OK

Orange moose = a portion that is defective

A random draw is an assurance activity & its result

Drawing a white bear (an activity that finds no defects) makes no change to the bag (quality of the software development artifact) but significantly increases our confidence in its contents (quality)

Sometimes a larger draw (more information) can leave you less confident(!)

Note: the better bag (better decision) may still be imperfect
Port & Wilf’s Software Release example

“Valuing Certification of Software Release Readiness,”
D. Port & J. Wilf (2013),
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)

Decision: send/don’t send MSL Rover Flight Software to System I&T?

- Case 0: Send and it passes (loss = 0)
- Case A: Send and it fails (loss = A)
- Case B: Hold it and it passes (loss = B)
- Case C: Hold it and it fails (loss = C)

Magnitudes: A > C > B

Assurance activity: JPL’s Software Release Certification Record (SRCR) process
If probability that software will pass, and values of A, B and C were known with certainty, then could compute which is the better decision (send/don’t send)

[akin to: if the contents of each bag were known with certainty, then could confidently compute which bag is better]
Port & Wilf’s Software Release example (cont.)

“Valuing Certification of Software Release Readiness,”
D. Port & J. Wilf (2013),
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)

In reality, probability that software will pass is uncertain (also A, B & C). JPL’s SRCR process involves assurance activities to answer questions; those answers decrease uncertainty. Decreased uncertainty improves chance of making the better decision. This can be valued – “the value of assurance”.

Lower “Expected Opportunity Loss” for “hold” with less overlap after assurance means we can make the “hold” decision with less decision risk.
Port & Wilf’s Software Release example (cont.)

“Valuing Certification of Software Release Readiness,”
D. Port & J. Wilf (2013),
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)

Use “credible intervals” as estimates of (un)certainty
e.g., expert judgment estimates $p = [0.75, 1.0]$

Bayesian updating:
• Given empirical (historical) data of how SRCR assurance questions and answers match pass/fail outcomes
• Determine how the assurance activity’s answers for this software change those credible intervals, e.g., was testing complete?
  • Yes: $p = [0.75, 1.0] \rightarrow [0.82, 1.0]$
  • No: $p = [0.75, 1.0] \rightarrow [0.75, 0.9]$

So the fundamental assurance question is:
how does an assurance activity’s outcome affect (un)certainty?
Assurance activities & findings w.r.t. Standard Processes

<table>
<thead>
<tr>
<th>Red = finding</th>
<th>Grey = no activity in this area</th>
<th>Does uncertainty here matter?</th>
</tr>
</thead>
</table>

Increased certainty! (+ defect corrected)

Overall: There are defects to be found, and assurance can find (at least some of) them

<table>
<thead>
<tr>
<th>What’s the value of assurance if it results in no findings?</th>
</tr>
</thead>
</table>

24
THANK YOU