Introduction

- M&S are being used more and more to support system design, development, evaluation and testing
- Most M&S used in DOD are “legacy” tools
  - Developed for one purpose, then used for many
  - Most have deficiencies in documentation, design & coding
- Wrong M&S results can cause ineffective systems, costly overruns, and even loss of life!
- Independent Verification and Validation (IV&V) reduces the risk of misusing these M&S
- We will show a number of lessons we have learned from IV&V of these tools
  - In support of a variety of programs
• VERIFICATION: The process of determining that a model implementation and its associated data accurately represent the developer’s conceptual description and specifications.
  – Does the model do what the originator intended
  – Is it relatively error free?

  **Did we build the model right?**

• VALIDATION: The process of determining the degree to which a model and its associated data are an accurate representation of the real world from the perspective of the intended uses of the model.
  – Do model results match real world data well enough for your needs?
  – Note that M&S validation is not the same as software validation

  **Did you build the right model?**

• ACCREDITATION: The official certification [determination] that a model, simulation, or federation of models and simulations and its associated data are acceptable for use for a specific purpose
  – Does the accreditation authority have adequate evidence to be confident that a model (and its input data) are credible and suitable for a particular use?

  **Is the Simulation Fit for this Purpose?**

Definitions from DODI 5000.61 dated 13 May 2003
GENERAL IV&V LESSONS LEARNED

• Focus on Intended Uses
  – Develop and test to M&S requirements related to defined uses

• Recognize that V&V is a continuing process
  – But know when to draw a line in the sand

• IV&V should concentrate on reducing the risk of using M&S results for the intended uses!
FOCUS IV&V ON M&S INTENDED USES

A lot of money is spent on V&V activities

• Cost-effective IV&V needs to focus on addressing:
  – What questions do the users need to answer?
  – What M&S outputs will be used to help answer those questions?
  – What characteristics must the M&S have to provide those answers?
    • Capability, Accuracy, Usability
  – What information is needed to show the M&S has those characteristics?
    • V&V results, CM artifacts, documentation, pedigree, etc.
  – What information is missing, and how can one best develop it?

• V&V should be tied to intended uses through requirements
  – Eliminate any M&S requirements that are not relevant to the specific intended use

• IV&V Team may need to help the user derive:
  – Detailed intended use statements
  – Requirements tied to those uses
It’s hard to conduct IV&V without software requirements

Example: IV&V of M&S that predicts Effective Time-on-Station (ETOS)*

Issue: The program had an intended use statement but no software requirements: this created many issues with verification as well as with the software

• The program office was changing requirements during M&S development
• The developer and the program office had no consensus on M&S requirements
  – A software requirements document serves as an agreement between the program office and the developer
• No software design requirements document meant there were no testable requirements for verification

Solution: We worked with the developer to create software design requirements

• These software design requirements were used as testable parameters to create an implementation test procedure
• Each requirement was matched with corresponding test(s)

*ETOS is defined as the total time the mission area is covered by an aircraft on station, divided by the total coverage time required
ETOS M&S IV&V Example

Independent V&V began with ETOS M&S Version 1:

• Verification test procedures were developed using the software requirements document we created
  – Multiple errors were discovered and documented
• Software Quality Assessment (SQA) was performed via manual code review
  – Relatively small code
  – Biggest issue was lack of objects
• Subject Matter Expert (SME) Review was performed
  – Verification errors were confirmed
  – Sensitivity analyses were reviewed
  – In general, SMEs agreed that M&S was realistic enough for the intended use if known errors were corrected
• The developer addressed multiple errors (bugs)
  – Developer can address software issues using the following justifications: User Error, Software Test Error, Software Requirements change, No Fix and Software Update
  – All major bugs should be fixed through a software update. Non major bugs can become new assumptions, limitations or known issues

Independent V&V continued with ETOS M&S Version 2:

• Verification test procedure was used again
• Multiple new errors were discovered
• Corrected in Version 3
**SME REVIEWS OF SENSITIVITY ANALYSES CAN SUPPLEMENT V&V**

Have the experts tell you if the results look suspect

- Example: ETOS should increase monotonically with endurance
- Sensitivity Analysis revealed a bug: M&S called back vehicle from sortie for scheduled maintenance, and re-launch would have been at night (deck closed for 12 hours); scheduled maintenance events should wait until sortie is completed
- That bug had not been identified by other V&V activities
MULTIPLE M&S FIXES MEAN MULTIPLE IV&V ITERATIONS

IV&V may never be completed if the M&S is in continuous use (and thus is continually evolving)

Independent V&V continued with ETOS M&S Version 3
- Verification test procedure was used a third time, using updated software requirements
- New errors were discovered; some older errors remained
- The developer addressed the most important errors

Version 4 consisted of fixing the newly discovered errors in V3
- Verification test procedure was used again
- The Software Requirements document and test procedures were updated with new requirements
- V&V Report was updated
- Results were determined to be adequate for use of Version 4 by the customer
  - Remaining issues were identified as limitations
YOU CAN’T AFFORD TO FIX EVERY PROBLEM YOU FIND

There’s no such thing as a perfect model

Major Bugs
• Show Stoppers that affect the intended use
  – All ETOS M&S V3 Show Stoppers were corrected in V4

Moderate Bugs
• Impact of error on intended use is understood and can be mitigated
  – These were not corrected in V4 but were identified as limitations
  – Normally only corrected if they can be done easily and cheaply

Minor Bugs
• Do not materially affect the intended use
  – Only corrected in V4 if they were very simple bug fixes
  – These were also identified as limitations
    – They are not limitations for the current use but could be for other uses
There is a proper order to good M&S development:
- Determine requirements
- Develop
- Test
- Update/Manage

Many of the errors discovered during ETOS M&S verification testing could have been avoided by creating the design requirements and having them approved by the program office before building the model.
- This would help avoid the issues created when the program office and the developer have different final products in mind.

Easy to say, hard to do in DOD
- Especially for continued use of “Legacy” M&S for a variety of purposes
- V&V is best done during M&S development, but in DOD it’s usually done after the fact
CREDIBILITY DOES NOT ALWAYS TRANSFER FROM OLD TO NEW VERSIONS

A Rose by Any Other Name…

• Many DOD M&S have evolved from older M&S
  – But may have been drastically changed in the process
• ETOS M&S
  – Had the same name as a previous code that had been used in the past
  – But the two actually had no code in common
• Issue: Program office was ready to accept the M&S based on previous use
  – But it wasn’t the same M&S at all!
  – Past uses of the older version added no credibility to this version
• Effective configuration management could have pointed out the problem
  – But it didn’t come up until IV&V was started
EFFECTIVE IV&V CALLS FOR A TEAM APPROACH

It works best when everybody pulls together

• Each team member brings different expertise:
  – Developer knows the specific M&S
  – V&V Team knows cost-effective V&V principles
  – SME know the technical area

• But they don’t always work and play well together
  – Developer can get defensive about his/her M&S
  – V&V Team can be perceived as hyper-critical
  – SME can get stuck on, “I’d have done it this other way”

• ETOS M&S example worked well:
  – Developer recognized the benefits of V&V improving the M&S
  – V&V Team objective was to help the M&S work for the user
  – SME made constructive suggestions based on review of sensitivity analyses
SOFTWARE DEVELOPERS DO V&V, THEY JUST DON’T WRITE IT DOWN

Documentation isn’t the fun part

• All software developers do a lot of work to convince themselves that their M&S works right
  – However they almost never document those results so they can convince someone else
  – Verification is best done during development, but usually is not adequately recorded

• Example: Mission Effectiveness Model
  – Developer initially said he had done no V&V and had no CM* process
  – However, we rummaged thru his files and found informal test reports, PowerPoint briefings and notes describing software test results going back over 20 years
  – He was able to trace versions throughout development, since he was the only developer and the principal user
  – We documented the results he had, added some V&V results of our own, and recommended a CM process that he implemented
  – The ultimate user accredited the M&S based on that evidence

* Configuration Management
THE FEDEP V&V OVERLAY DOESN’T ALWAYS FIT VERY WELL FOR DISTRIBUTED SIMULATION

One size does not fit all…

• Two options for V&V of distributed M&S in DOD:
  – Treat the M&S as a black box and only look at I/O values
  – V&V Overlay to the FEDEP* (standard for HLA)

• The customer insisted we use the FEDEP V&V Overlay for a distributed M&S, but it didn’t work out very well
  – It wasn’t HLA, and the architecture it used did not mandate the same documents (Federation Objectives, Conceptual Model, Design Documents, Federation Development Documentation, etc.)
  – So we didn’t have the information needed to do V&V, and there wasn’t time for the program to generate it
  – We ended up just making sure the pieces could talk to each other
    • But didn’t have time to make sure that what they said to each other made sense

• The lesson here is that the FEDEP overlay was the wrong approach
  – Distributed simulations take more time and effort to do V&V that way
  – For that approach to work, V&V needs to be coordinated across the distributed players early on in development

*Federation Development and Execution Process
V&V is a “rheostat” gradually shining light on the problem: how much light you need depends on the risks of using M&S results

- The ultimate goal of IV&V efforts is to form a foundation for making good program decisions
  - Credible, well understood tools allow a sound analysis which leads to good program decisions

- Nature and extent of information required to support accreditation decision is at the discretion of the accreditation authority and is generally based on an assessment of risk
  - Role of M&S results in decision making process
  - Importance of decision that M&S is supporting
  - Severity of the consequences of making incorrect decisions because M&S results were wrong
  - Likelihood that M&S results are wrong
Reducing the Risk of M&S Use

Risk = Likelihood x Impact

IV&V Results

Reduce M&S Credibility

Risk Reduction Strategies

Reduce reliance on M&S results
## Correlating Risk with V&V Activities

<table>
<thead>
<tr>
<th>M&amp;S S/W Accuracy Issue</th>
<th>Items Required</th>
<th>Typical Sources</th>
<th>Type, Scope and Depth of Information Required When Risk Is...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S/W development and maintenance process description</td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>How much confidence do you have in the accuracy of the software?</td>
<td>S/W development and management resources description</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S/W development and management artifacts and documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S/W verification results</td>
<td>Module, subsystem and system S/W test reports; S/W Problem Change Request (SPCR) logs that correlate verification results with specific versions of the S/W; alpha- or beta- test reports; specific verification reports for the M&amp;S version being used. IV&amp;V reports</td>
<td>System level verification test results desirable. System and subsystem level verification test documentation is required. System, subsystem and module level verification test documentation is required. IV&amp;V results are desirable.</td>
</tr>
<tr>
<td></td>
<td>S/W Quality Assessment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Example from Accreditation Information Requirements Guide (AIRGuide)
- AIRGuide covers M&S Capability, Accuracy & Usability Issues
- AIRGuide requirements are based on interviews with 40+ DOD Programs
The Essence of Accreditation

**TO PROVE THE M&S IS FIT FOR PURPOSE:**

REQUIRES AN OBJECTIVE COMPARISON
OF M&S REQUIREMENTS WITH M&S INFORMATION
WITHIN THE CONTEXT OF THE PROBLEM
TO ASSESS THE **RESIDUAL RISK** OF USING THE M&S
SUMMARY AND CONCLUSIONS

• All IV&V lessons stem from two “cardinal rules”:
  1. IV&V activities must be driven by the requirements of the M&S intended uses
  2. IV&V activities must be designed to reduce the risk of using the M&S for those intended purposes

• IV&V activities not focused on reducing the risk of specific M&S uses are wasting your time and your customers’ money

• Accreditation Authority must answer the question: “Do I know enough about how well the M&S works to use it with acceptable risk?”
Key Issues for a Successful SME Review

- The type of SME Review must be well defined:
  - Face Validation Review
  - Design Review
  - Accreditation Review
- Review purpose must be clear
  - Identify M&S intended use, acceptance criteria and assessment methods
  - Read-ahead package is helpful
- Records must be kept
  - Participants’ qualifications
  - Record of discussion and conclusions
- Participants must be carefully chosen
  - Representing a cross-section of users and subject experts
- Review must be structured, yet allow for discussion
  - And the report must be prepared in a timely manner
VV&A as Risk Reduction

• VERIFICATION
  – Reduces the risk that the software you build (or use) has undetected errors in it that are fatal to your intended use
  – Reduces the risk that the data is inappropriate for the intended application or improperly prepared

• VALIDATION
  – Reduces the risk that simulation outputs won’t match the “real world” well enough for you to use them credibly as part of the solution to your problem
  – Reduces the risk that the data doesn’t represent the real world with sufficient accuracy for the application

• ACCREDITATION
  – Reduces the risk of making a wrong decision because an inappropriate or unsuitable simulation is selected for use in solving a problem
Ideal Development Steps for Verification

1. Developers and Users develop the requirements which system must meet.
2. Software Requirements Specification (SRS) is written to define those requirements and agreed by all parties.
3. Design Verification Test Procedure is created with the tests that will be used to ensure that the software meets the agreed requirements. *(Can be done later in development process as well. Sometimes it makes the programming easier in many cases to have tests to use as you go along.)*
4. Software development is started.
5. When either the developer or the user changes a software requirement it is agreed and added to the SRS. A new version of the SRS is released.
6. Each requirement is tested internally with documented results. Errors found are recorded and tracked.
7. For very sensitive systems another round of testing is completed by an outside Independent Verification and Validation group (IV&V).
8. If the program office plans to use verification results to add to credibility, results of verification testing MUST be documented.

*When steps are skipped it slows down the development process.*
Verification is interested in the testable software requirements. If a requirement is not written to be testable it cannot be verified. Many times this involves including metrics in more generalized requirements.

**Example:** Testable Requirement: The system shall maintain a frame rate of at least 60 FPS at all times

Non-Testable Requirement: The system shall maintain a high frame rate

A requirement must have a testable metric in order to be verified in this case. “High Frame Rate” while it sounds good from a nontechnical perspective, it is useless for verification without the metric of 60 FPS.
Testable Requirements

There must be a way to test each requirement. Sometimes verification modes and settings may have to be added to a program to verify subsystem requirements. Or different parts can be tested separately.

Some programs or systems can be quite large with many different parts, both hardware and software. There can be separate requirements for each subsystem and the overall program. There may be lots of different parts and interfaces. When the requirements are specified, each part may have its own verification and requirements.

Any requirement that requires a measurement must have a tolerance. If it does not then it must be exact to the least significant number. **This is very important for hardware measurements.**

EX: Different ways to write 1.00 ± 0.05

Correct: 1.00 ± 0.05V, 1.0V
Incorrect: 1.00, 1 ± 0.05V, 1.00
Testable Requirements

We summarize verification as "Are you building it right?"

Verification involves producing documented evidence that a system performs in accordance to the software’s requirements.

If a program’s requirement does not define what a system does, the developer can say anything they are building is correct. Therefore, the first step in any software-development effort should be to define what is being built.

*When one defines what they are building in advance, verification is simply testing that the software matches that definition!*

All requirements should be numbered. The number of the requirement should not change or be used for another requirement in the future.

Once numbered, all requirements should keep the same number throughout all the versions of the software-requirements document. If a requirement is removed a new requirement should not be given the same number.

*This is very important for regression testing and traceability. It will save you a lot of time and headache.*
Software Requirement Change

When a requirement must be changed for any reason, the requirements document is simply updated and a new version is released.

All parties with interest in the program must agree to these changes; if the developer cannot meet a requirement they need to sit down with the project manager and create a requirement that is more reasonable. Or, if the program office wants added functionality, they have to sit down with the developer and propose a change of requirements.

A new version of the requirements should be released for each such update. Each requirement changed should be documented in the ‘change log’ of the requirements document.
Design Verification Test Procedure

With well-written requirements, tests are very easy to write.

Tests can be written before the software is even created.

The test is written simply to verify the requirement, using the metrics defined in the requirement.

The test procedure should be reusable and the tests should be reproducible.

All tests should include at a minimum the following.

- Test Number
- Requirement Being Tested
- Test Steps
- Expected Result for each step
- Actual Results for Each Step
- Pass/Fail
Design Verification Test Procedure

All verification measurements must be completed with calibrated measurement devices with sensitivities appropriate for the measurement being made. The calibration date and product description should be recorded in the test procedure.

All tests should be written so that the least skilled person on the test team can run the test and reproduce the results. If you are going to be bringing in an IV&V team then the tests should be written so that they can run them on their own without the developer’s input.

Always test boundary cases. So if the system is required to complete a procedure in 1 second put the system under the maximum possible load and record the runtime.

Sometimes this will reveal that some of the requirements have unrealistic input and output parameters. The change might simply be to redefine the requirement and write a new test. (Then to update requirements document and get approval)

All tests should include the requirement it is testing. A test can test more than one requirement, or their may be more than one test for a requirement.
Measurements

All verification measurements must be completed with calibrated measurement devices. If an instrument has not been calibrated within the last year, it should not be used for a very sensitive measurement.

*All measurements should be taken with appropriate equipment. If the requirements state that an output must be 1.000 mA ± 0.002 you must use equipment that can take very small amperage measurements. If the true result was 1.003, there is a lot of measurement equipment out there that will give you 1.0 as a result.*

The calibration date and product description should be recorded in the test procedure.

This is very important for accuracy and capability. If these procedures are not followed for very sensitive outputs and inputs, it should be noted in the log.
Writing Requirements after the fact: NOT IDEAL & VERY BAD!!!!

It has been brought to our attention that many software programs do not have requirements. **If a program has no requirements it cannot be verified.** It should be given a rating of maximum risk in the verification section. So the first step is to help the program office create requirements.

It is very hard for an outside group to write requirements after the fact on a software program. **NORMALLY THIS SHOULD NOT BE DONE.** it requires the outside group to completely reverse-engineer the entire software program, but if you have to do so, here are some steps I suggest:

The V&V team might have to write the requirement from what the software does instead of what it is intended to do.

The V&V team may have to make sure that the program office approves every requirement that they write for the program.

a. There is a chance that a requirement the V&V team wrote a requirement that is actually a bug! Just because a program functions in a certain way doesn’t mean that is what was intended.

b. **You are not the one who approves any requirement!** Don’t take the blame when a program doesn’t work how the user intended because you wrote a bad requirement for their program. Make them approve every single requirement; this is the #1 most important point.

c. The V&V team can’t write requirements based on what a program does to verify it because then they will never find any errors--- which is the point of verification.
While writing requirements after the fact your main focus is to get the program verified. Every requirement you write should have a test you can run in mind. Don’t waste time with anything else.

If the program office or developer tries to skimp on requirements in order to make the program pass verification, that’s fine so long as the program office approves the requirements.

It is not part of verification for you to tell them what their simulation is required to do.

- If their requirements state an airplane can only makes left turns at 10000 feet in the air on sunny days, and the simulation does that, it passes verification.
- Whether or not this is useful is a question for the experts in the field and is done during the validation.
- If they agree to the requirements and those requirements make the simulation useless, they will be the ones who look bad, not you.
- They will also be stuck with a simulation that is not verified for anything but making right turns at 10000 feet on a sunny day. This must be noted in the capability section of the V&V report.
The IV&V testers do not need to write the tests. They can run the tests created by someone else.

All test results must be recorded and labeled ‘pass’ or ‘fail’.

JUST BECAUSE A REQUIREMENT FAILS VERIFICATION DOES NOT MEAN THAT A PROGRAM IS BAD.

Software failures should be documented and recorded. Then they should be addressed by the software team and program office. This does not mean that they need to be fixed; here are the possible ways to address a verification test failure:

All tests should be given a level of severity by the tester. This is just an estimate and can be changed by the developer and the program office at a later date.

All documented failures should be linked to a requirement number and the test in which it was found (if found during a test).
Tester Error: The tester made a mistake and upon retest the issue was found to be a tester’s error.

This happens often; no one is perfect.

On an IV&V team, many times the tester does not have the expertise of the developer in using the software and can make mistakes. This is not a big deal and should not ever be used for an excuse for not documenting something that the tester thinks is an error! Many new people in verification at the places I worked are afraid to record errors because they think they made a mistake running a test.

IT’S MUCH WORSE TO MISS AN ERROR THAN TO MISIDENTIFY SOMETHING AS AN ERROR.

A tester may see something that looks like an error that has nothing to do with the test. This is the most common error and is good. You should record this potential error even if it’s not part of the exact test that is being worked. DO NOT IGNORE IT!!!!!! The justification for closing these can be either ‘working as intended’ or by stating what the tester wrongly did.
Failure Resolution Examples

**NO FIX**: This is very common. The error has been identified and many times this error only occurs with unrealistic input or output parameters. The issue is documented and a justification for the ‘no fix’ is produced. In a simulation, you have to test to make sure the error does not occur using reasonable parameters, or the error is so small it does not affect the intended use.

Example: At GE on the CT scanners’ user interface (UI) the maximum power setting was below the requirement which was set to limit the maximum dose someone receives. On the UI the scanner could not use that high of a power setting, so the requirement failed verification. This was labeled a no fix going into the next round of testing because this caused the system to be MORE safe than required. The requirement was not changed at this time because in the future this requirement could be met, but it didn’t affect anything else.
**Failure Resolution Examples**

**Software Update:** The error is addressed by a new software release.  
If an error is serious enough, a new version of software is released. Verification is an iterative process that sometimes requires many rounds of testing and software releases. If the software had good requirements and tests written for it, then most of these errors should have been found by internal verification testing before any IV&V team works on the project.

**Requirements Change:** The verification testing failure is caused by an error in the requirements document, or the requirements are too inclusive.

Example: This happens a lot at my last job with measurements. If you make a requirement that a resistor on a component be 100 ohms to within ±0.001 ohm---sometimes the components chosen for cost reasons are within ±0.01 ohms. Well, that is good enough for the intended use, so the requirement is updated and the test rerun.
Failure Resolution Examples

Test Error: The test is failed because it does not properly test the requirement. This happens a lot when requirements have changed, but the test document was not updated. Simply write in the justification how the test was changed and close the issue. This should only be done if it’s a legitimate testing error. You shouldn’t change a test just to make your program pass verification.

Intermittent issues: Sometimes a program crashes or another issue comes up with no apparent cause, and it cannot be reproduced reliably. The tester should gather and document as much data as they can on the issue. This issue could show up again, and with more information it could help reveal what is causing it. These are the hardest bugs to deal with and should not be ignored.

Sometimes issues are left as ‘no fix’ and left open and documented as limitations. They may or may not be fixed in later software releases.
A system should not list capabilities that the requirements do not support. The requirements should match the expected capability and intended use. If it does not, then the developer or the program office has to change their expectations of the system’s performance.

**Limitations** can be limitations from software errors or the fidelity defined in the requirements

**Known Errors** can be derived from the bugs list after verification. Even if these errors are not fixed and if you know about them, you can use that information to make better decisions. If the error does not affect the intended use and are very minor, they do not always have to be listed.
Accuracy

Writing requirements and subsequent verification testing define exactly how accurate the model will be in advance, and also will define the fidelity and functionality of the model.

Verification does not ensure that the model will match the real-world results. It only ensures that the model functions as defined in the requirements. If you build the model using inadequate requirements, you will still have an inaccurate model.

Verification test results will add credibility to a model by providing evidence that it functions in the way defined by the requirements.

If a model is verified to work as intended, the requirements can be used by subject matter experts as part of the bases for determining that the model properly simulates the real world during validation.

Validation without Verification can lead to a bad model being called good. Just because the models algorithms are correct and work under a few cases does not mean they are implemented correctly for all cases!!!
Usability

Manuals and users’ materials should be reviewed and verified to be correct.

Users should be aware of issues found during verification and the resulting workarounds. If an error is known, the results of the simulation can be used even if the model is not fixed. Serious problems occur when errors are not known.
Relationship to Verification Principles*

• Principle: The goal of solution verification is to estimate, and control if possible, the error in each quantity of interest (QOI) for the problem at hand.
  – Develop Sensitivity Analyses focused on the M&S Intended Uses

• Principle: Solution verification is well defined only in terms of specified QOI, which are usually functionals of the full computed solution.
  – Conduct Sensitivity Analyses of Outputs of Key Interest to those intended uses

• Principle: The efficiency and effectiveness of code and solution verification can often be enhanced by exploiting the hierarchical composition of codes and mathematical models, with verification performed first on the lowest-level building blocks and then on successively more complex levels.
  – Utilize both end-to-end and functional level sensitivities in the analysis

• Best Practice: Subject-matter expertise should inform assessments of relevance.
  – Conduct SME Review of sensitivity analysis results

• Principle: Verification is most effective when performed on software developed under appropriate software quality practices.
  – True, but often hard to find in our experience, and documentation always seems to lag way behind code development

* Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation and Uncertainty Quantification, Committee on Mathematical Foundations of Verification, Validation, and Uncertainty Quantification, Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, National Research Council of the National Academies, The National Academies Press (pre-publication draft) pp 7-1, 7-2, 7-4
M&S VV&A

• What is “risk?”
  – How do you define “risk” for M&S use?
  – Do you use risk to determine VV&A strategy?
    • If so, how?
    • If not, what do you base your strategy on?

• How do you decide to accredit an M&S?
  – What makes an M&S credible?
    • Capability, Accuracy, Usability?
  – Is V&V always necessary?
    • Why isn’t V&V always done?
    • Who should do V&V?
  – What do you do if you can’t get validation data?
  – Who decides you’ve done enough, and how?
  – If it’s already been accredited by someone else, why do more?