SANDIA REPORT

SAND2007-0570

Printed December 2007

Modeling and Simulation Technology Readiness Levels

Robert L. Clay
Scot J. Marburger
Max S. Shneider
Timothy G. Trucano

Prepared by Sandia National Laboratories
Albuquerque, New Mexico 87185, and Livermore, California 94550
Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

SANDIA REPORT
SAND2007-0570
Printed Month Year

Modeling and Simulation Technology Readiness Levels

Robert L. Clay
Scot J. Marburger
Max S. Shneider
Timothy G. Trucano

ABSTRACT

This report summarizes the results of an effort to establish a framework for assigning and communicating technology readiness levels (TRLs) for the modeling and simulation (ModSim) capabilities at Sandia National Laboratories. This effort was undertaken as a special assignment for the Weapon Simulation and Computing (WSC) program office led by Art Hale, and lasted from January to September 2006. This report summarizes the results, conclusions, and recommendations, and is intended to help guide the program office in their decisions about the future direction of this work.

The work was broken out into several distinct phases, starting with establishing the scope and definition of assignment. These are characterized in a set of key assertions provided in the body of this report. Fundamentally, the assignment involved establishing an intellectual framework for TRL assignments to Sandia's modeling and simulation capabilities, including the development and testing of a process to conduct the assignments. To that end, we proposed a methodology for both assigning and understanding the TRLs, and outlined some of the restrictions that need to be placed on this process and the expected use of the result. One of the first assumptions we overturned was the notion of a 'static' TRL – rather we concluded that problem context was essential in any TRL assignment, and that leads to dynamic results (i.e., a ModSim tool's readiness level depends on how it is used, and by whom). While we leveraged the classic TRL results from NASA, DoD, and Sandia's NW program, we came up with a substantially revised version of the TRL definitions, maintaining consistency with the classic level definitions and the

Predictive Capability Maturity Model (PCMM) approach. In fact, we substantially leveraged the foundation the PCMM team provided, and augmented that as needed.

Given the modeling and simulation TRL definitions and our proposed assignment methodology, we conducted four 'field trials' to examine how this would work in practice. The results varied substantially, but did indicate that establishing the capability dependencies and making the TRL assignments was manageable and not particularly time consuming. The key differences arose in perceptions of how this information might be used, and what value it would have (opinions ranged from negative to positive value). The use cases and field trial results are included in this report. Taken together, the results suggest that we can make reasonably reliable TRL assignments, but that using those without the context of the information that led to those results (i.e., examining the measures suggested by the PCMM table, and extended for ModSim TRL purposes) produces an oversimplified result – that is, you cannot really boil things down to just a scalar value without losing critical information.

Table of Contents

List of Figures

List of Tables

Executive Summary

This report summarizes the results of an effort to establish a framework for assigning and communicating technology readiness levels (TRLs) to the computational science and engineering (CSE) modeling and simulation (ModSim) capabilities at Sandia National Laboratories. (We always intend 'ModSim' in this report to mean 'CSE ModSim.') This effort was undertaken as a special assignment for the weapon simulation and computing (WSC) program office led by Art Hale, and lasted from January to September 2006. This report summarizes the results, conclusions, and recommendations, and is intended to help guide the program office in their decisions about the future direction of this work.

The work was broken out into several distinct phases, starting with establishing the scope and definition of assignment. These are characterized in a set of key assertions provided in the body of this report. Fundamentally, the assignment involved establishing an intellectual framework for TRL assignments to Sandia's Advanced Simulation and Computing (ASC) program ModSim capabilities, including the development and testing of a process to conduct the assignments. To that end, we have proposed a methodology for both assigning and understanding the TRLs, and outlined some of the restrictions that need to be placed on this process and the expected use of the result. One of the first assumptions we overturned was the notion of a 'static' TRL – rather we concluded that problem context was essential in any TRL assignment, and that leads to dynamic results (i.e., a ModSim tool's readiness level depends on how it is used, and by whom). While we leveraged the classic TRL definitions from NASA, DoD, and Sandia's NW program, we came up with a substantially revised version of the TRL definitions, maintaining consistency with the classic level definitions and the Predictive Capability Maturity Model (PCMM) approach being developed by the SNL ASC Verification and Validation (V&V) program. In fact, we substantially leveraged the foundation the PCMM team provided, and augmented that as needed.

Given the modeling and simulation TRL definitions and our proposed assignment methodology we conducted four 'field trials' to examine how this would work in practice. The results varied substantially, but did indicate that establishing the capability dependencies and making the TRL assignments was manageable and not particularly time consuming. The key debate arose in perceptions of how this information might be used, and what value it would have (opinions ranged from negative to positive value). The use cases and field trial results are included in this report. Taken together, the results suggest that we can make reasonably reliable TRL assignments. However, using those assignments without the context of the information that led to them (i.e., examining the measures suggested by the PCMM table, as extended for ModSim TRL

purposes) produces an oversimplified result – that is, you cannot really boil things down to just a scalar value without losing critical information.

There are four main conclusions and associated recommendations from this report, discussed in detail in the final section of the main body of this report. They are:

- **Conclusion 1**: We can assign TRLs to ASC ModSim capabilities.
- **Recommendation 1**: Use the framework and process proposed in this report as a baseline for refinement.
- **Conclusion 2**: ModSim TRLs and the PCMM specification are connected.
- **Recommendation 2**: Conduct TRL assessments as an adjunct to the PCMM process, not as a stand-alone exercise.
- **Conclusion 3**: The ModSim TRL framework and process described in this report are not complete – there is still work to be done.
- **Recommendation 3**: If the sponsors decide to proceed with this line of development, continue to resolve the framework and process using the baseline approach described in this report.
- **Conclusion 4**: There remain questions about the utility of ModSim TRLs.
- **Recommendation 4**: Address key issues before proceeding to the next phase of developing the TRL framework and process.

Introduction

This report summarizes the results of an effort to establish a framework for assigning and communicating the technology readiness levels of the modeling and simulation capabilities at Sandia National Laboratories. This effort was undertaken as a special assignment for the SNL Weapon Simulation and Computing (WSC) program office led by Art Hale, and lasted from January to September 2006. This report summarizes the results, conclusions, and recommendations, and is intended to help guide the program office in their decisions about the future direction of this work.

The Scope of the ModSim TRL Effort

Our charter was to develop a framework for the assignment and communication of TRLs for Advanced Simulation and Computing (ASC) program computational science and engineering (CSE) modeling and simulation (ModSim) capabilities at Sandia National Laboratories. (In what follows, the acronym 'ModSim' will always means 'CSE modeling and simulation' unless specifically noted.) Since this had not been done before at Sandia and little implementation information about applicability of TRLs to CSE modeling and simulation exists at other DOE institutions, there was an exploratory aspect to this work. At the outset, we established some guiding principles that were cast as assertions and vetted by a steering committee (which was, in fact, more of a working group). This steering committee included the following people: Robert Clay (chair, ModSim TRL team), Paul Yarrington (WSC program office manager), Timothy Trucano (QMU/V&V staff), Mike Hardwick (ASC CSSE manager), Pete Wilson (engineering analyst manager), Mike Chiesa (engineering analyst manager), Fran Current (WSC program office manager), Scott Klenke (DSW staff), Scot Marburger (ModSim TRL team), and Max Shneider (ModSim TRL team). This group was intended to represent a reasonable cross section of the WSC stakeholders for the current effort.

Key Assertions Used to Guide this Effort

During the scope-definition phase of this effort we developed a set of assertions that established some baseline principles for the work to follow. These were reviewed by the steering committee (described above), and captured as follows in a set of eleven core assertions that define the scope and purpose of this work.

Assertion 1: ***ModSim capability is the ability to simulate weapons systems physical and engineering performance in a specific context using CSE modeling and simulation tools, expertise, and computing hardware.***

This assertion is a straightforward definition of what we mean by 'modeling and simulation capability' in the context of NWSMU ModSim. It was drawn from various sources, including numerous interviews with a cross-section of individuals representing the capability providers (tool developers), users (analysts), and consumers (SNL nuclear weapons engineers).

The notable extensions to this definition are the notion that a ModSim capability is not just the tool (software/code), but also requires consideration of user qualification[1] (expertise) and infrastructure (computing hardware) to be complete.

Assertion 2: ***We are determining ModSim capability readiness because it is the right thing to do.***

This statement goes directly toward answering the question of "Why are we doing TRLs?" There are potentially numerous uses and purposes for establishing a ModSim TRL framework, but the ones called out by the steering committee include the following:

- communicate ASC preparedness to respond to customer needs
- communicate the maturity level of ASC ModSim products and the confidence that can be placed in the technology
- help guide ASC programmatic investments in analytical capabilities.

This is further extended in Appendix 7, the WSC Program Office Use Case, provided by Paul Yarrington and refined by the authors.

Assertion 3: ***Our sponsor is the WSC management team, and our stakeholders include:***

- ***PCMM team (also partners)***
- ***NWSMU TRL team***
- ***Weapons Engineers***
- ***Analysts***

[1] The point of user qualification was initially questioned, but think of handing a hammer and chisel to Michelangelo and a chimpanzee, each with a slab of marble and instructions to sculpt 'David', and you clearly get the sense that who is using the tool matters in terms of the expected outcome. Implicitly assuming that experts are always using ModSim tools, so as to factor the expertise issue out of the TRL specification, is not a valid approach because the impact of expertise on ModSim predictions is profound.

- ***ModSim developers***
- ***ASC program office.***

This is simply a statement of who our sponsor and stakeholders are. As part of this discussion, we established a basic requirement that the ModSim TRL framework needed to be consistent and compatible with the ASC V&V/PCMM program effort and the NWSMU TRL approach for hardware products and capabilities. This is also explicitly stated in Assertion 7.

Assertion 4: ***"Static" TRLs do not work for ModSim capabilities, since the readiness level depends on the question and context. Therefore, we need a flexible approach to assessing ModSim TRLs that is responsive to problem context.***

We were initially hoping to establish the TRLs in core capability areas for ModSim. However, it quickly became apparent (see discussion in Assertion 1 above) that it is essential to establish the problem/usage context prior to assigning a TRL. The reason for this is readily apparent upon consideration – how well a tool will perform depends on what you intend to do with it. When you couple this basic notion to the dependency on who is using the tool and where they are using it, the notion of a static (i.e., non-time-dependent) TRL assignment becomes illogical. The steering committee did acknowledge that high-level TRL assignments could be made for generalized capabilities so long as the set of assumptions for those assignments were clearly documented (e.g., what is the set of problems of interest, what split (weighting) between the various problem types is necessary to form an aggregate TRL, etc.).

Assertion 5: ***Independent assessments improve credibility.***

A requirement placed on the team from the WSC program office was to define a robust, relatively objective approach to TRLs – that is, for a given set of conditions (problem, persons, tools), the answers should be consistent and stable (meaning that the same problem, persons, tools should yield the same TRL evaluation). This assertion simply emphasizes the need for independence in the TRL assignment process, and the methodology we propose in this report calls for independence in the assignment teams. 'Independence' specifically means 'independent of the tool/capability developers.'

Assertion 6: ***The ModSim community includes:***

- ***producers (tool/capability developers)***
- ***users (analysts)***
- ***consumers (product engineers)***

- ***sponsors (WSC/ASC).***

This assertion simply states explicitly who we view as the modeling and simulation community associated with this ModSim TRL framework.

Assertion 7: ***Our TRL solution needs to be compatible with the ASC/PCMM and NWSMU TRLs.***

By 'compatible,' we mean that the TRLs for ModSim must be able to be unambiguously communicated within a framework involving these stakeholders.

Assertion 8: ***There will be nine TRLs to conform to existing NWSMU convention. We will define the levels for ModSim, and then map that back to NWSMU TRLs.***

This requirement simply states that there will be nine levels for ModSim TRLs, as is common practice for hardware TRL assignments at NASA, DoD, and Sandia's NWSMU. We note that the quantization of the TRL process directly mirrors the fidelity of the process for performing the assessment. Nine evaluation levels requires a process that can unambiguously resolve nine different levels of information required in TRL evaluation.

Assertion 9: ***We want to assign readiness levels to ASC and commercial tools, hardware, and expertise.***

This statement acknowledges the fact that we are interested in ModSim capabilities, whether the source is ASC or commercial suppliers. From the customer's perspective, the issue is how well the tool performs the job, not who provides the tool. (Obviously, evaluating commercial ModSim tools external to the SNL ASC program poses some different problems than for ASC tools, such as availability of needed V&V information.)

Assertion 10: ***A rigorous, reproducible process is required.***

This is related to Assertion 5 above. Two key points were brought out during this discussion of this assertion:

- Consistency in the assessment process is required to enable TRL comparisons across time.
- Consistency in the assessment process also ensures that comparisons between similar capabilities are meaningful.

Assertion 11: ***We are working under the assumption that some of the information involved in the TRL assessment process will be classified, and therefore we may need a classified version of the process.***

This simply acknowledges that our framework and process need to be able to address classified information and capabilities.

ModSim TRL Definitions and Their Mappings

In order to make TRL assignments, we first need to define how many levels there are and what we mean by any given level. Standard examples of TRL definitions exist from NASA, DoD, and Sandia's NWSMU, all of which are primarily oriented at defining TRLs for hardware. Sandia's ModSim capability set is an entirely different category of technology from hardware. First, a ModSim capability is generally not a physical device, but a synthesis of software, hardware, and expertise (see Assertion 1 above). Second, there is no fixed specification, operating environment or 'finished product', per se. Most of the software tools are evolving on an ongoing basis and their application (specification and operating environment) can change often. Basically, they are moving targets in general, and in many cases the core components (features in the codes and pre and post processing tools) involve an R&D component whereby new features show up in an ongoing manner. Fundamentally, the concept of 'readiness' for ModSim is uncertain. This is entirely different from the standard TRL model, and presents a host of issues for the application to modeling and simulation.

Our initial attempt to create ModSim TRL definitions was based on a modest transformation of the language in the NASA, DoD, and Sandia NWSMU 'hardware-based' TRL definitions. In four field trials with this set of definitions the participant teams found the language vague and, in general, not an adequate representation of the key attributes associated with ModSim capability readiness. Further, there was no clear mapping between that language/scheme and the PCMM maturity table[2]. As a result, we revised the definitions by leveraging the PCMM attributes and language, and recast the definitions as a table where the rows are TRL levels and the columns are augmented PCMM key attributes (see Tables 1 and 2). This change of the definitions had the advantage of being relatively consistent with the PCMM approach (since we directly used much of their language and measures) and completely consistent with the NWSMU TRL levels (i.e.,

[2] M. Pilch, T. Trucano, and J. Helton, SAND2006-5001; more extensive documentation of the PCMM (by Pilch, Oberkampf, and Trucano) is in progress at the time of writing. Some additional comments on measuring ModSim capability dimensions are found in Trucano, SAND2006-7725P

both approaches used nine levels, and the 1-to-1 mapping is consistent based on our test assessments). Another round of field tests indicated that this new configuration was much better suited to the task and preferred by the assignment teams. These tests also provided a number of refinements to the definition table, most of which have been incorporated into Tables 1 and 2.

TRL	Capability Maturity	Verification[3]	Validation	User Qualification
1	Concept Phase: basic principles identified.	Judgment only, or numerical errors unacceptably pollute validation or application decisions.	Judgment only. Insignificant coverage of the dominant physics. Dominant physics assessed to be inadequate.	None required.
2	Concept Phase: technology concept and/or app formulated.			
3	Concept Phase: proof of concept initiated.			
4	Prototype Phase: concept demonstrated on 'toy'/lab problem.	Explore sensitivity to discretization and algorithm parameters.	Qualitative comparisons of measurement to predicted. Substantially incomplete coverage of dominant physics.	Familiar with similar tools on similar problem type.
5	Prototype Phase: key elements demonstrated on realistic problem.			
6	Prototype Phase: system model demonstrated on realistic problem.	Estimate numerical errors.	Quantitative validation w/o assessment of variability and uncertainties in diagnostics and model. Or, w/ significant extrapolation to application parameter space. With significant coverage of dominant physics.	Familiar using this tool on similar problems and computer systems.
7	Production Phase: system demonstrated on realistic problem in production.			Familiar using this tool on similar problems and production computer systems.
8	Production Phase: system completed and qualified on production through test and demonstration.	Quantify rigorous numerical error bounds.	Quantitative validation w/ assessment of variability and uncertainties in diagnostics and model. Without significant extrapolation to application parameter space. With significant coverage of dominant physics and their interactions.	Routine production use of tool on similar problems on the target production computer system.
9	Production Phase: system completed and in ongoing production use.			

Table 1 – Modeling and Simulation TRL Definition Table (Part 1)

[3] Description for verification and validation are taken from the PCMM table by Pilch, Trucano, and Helton, SAND2006-5001.

<table>
<tr><th>TRL</th><th>Code Readiness[4]</th><th>Models[5]</th><th>Geometry[6]</th><th>QMU[7]</th><th>System[8]</th></tr>
<tr><td>1</td><td rowspan="3">Judgment only. Critical features and capabilities are missing or lack robustness. Sustained unit/regression testing w/o significant coverage. Unsustained unit/regression testing w/ or w/o significant coverage.</td><td rowspan="3">Model form unknown.</td><td rowspan="4">Low fidelity: Significant defeaturing and/or simplification of geometry. Low level of detail represented (e.g., block representations of assemblies and parts).</td><td rowspan="3">Deterministic Best Estimate or nominal margins. Judgment-only assessment of uncertainty and sensitivity.</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
<tr><td>4</td><td rowspan="2">Code managed and assessed against SQE requirements. Sustained unit/regression testing w/ significant coverage. Unsustained verification testing w/ or w/o significant coverage.</td><td rowspan="2">Empirical model forms speculated or calibrated to represent trends. Calibration of physics-informed models.</td><td rowspan="2">Deterministic margins. Or, informal “what if” assessment of uncertainty and sensitivity.</td><td>4</td></tr>
<tr><td>5</td><td>5</td></tr>
<tr><td>6</td><td rowspan="2">Code managed and assessed against SQE requirements. Sustained unit/regression testing w/ significant coverage. Sustained verification testing w/ significant coverage of separate physics.</td><td rowspan="2">Alternate plausible physics-informed models. Potentially w/ model form calibration.</td><td rowspan="3">Initial attempts to formally quantify margins, uncertainty, and sensitivity. With significant judgment, or significant judgment as to what to include.</td><td>6</td></tr>
<tr><td>7</td><td rowspan="3">High fidelity: Geometric representation consistent with “as built”, with little to no defeaturing and/or simplification. Appropriate level of detail for qualification. Small features and parts captured.</td><td>7</td></tr>
<tr><td>8</td><td rowspan="2">Code managed and assessed against SQE requirements. Sustained unit/regression testing w/ significant coverage. Sustained verification testing w/ significant coverage of high-order interactions.</td><td rowspan="2">Established physics-based model.</td><td rowspan="2">Formal quantification of margins, uncertainty, and sensitivity. Without significant judgment as to what to include.</td><td>8</td></tr>
<tr><td>9</td><td>9</td></tr>
</table>

Table 2 – Modeling and Simulation TRL Definition Table (Part 2)

[4] Description for software attributes (columns) are taken from the PCMM table by Pilch, Trucano, and Helton, SAND2006-5001. Geometry attribute description altered to focus more on fidelity instead of dimensionality.
[5] Physics and material models – applies to simulations.
[6] Physical geometry – applies to simulations.
[7] QMU and sensitivities.
[8] Computer system TRL.

Further explanation of the entries and usage model for this combined table is in order. We will start with explaining the entries in more detail than the table itself provides. The first column in both tables is the numerical value of the TRL (i.e., the row labels).

Capability Maturity: This attribute is a condensation of the traditional TRL definition, and is meant to give a gross numerical indicator of the general state of the capability in question. Users decide whether it is in the concept, prototype, or production phase, and then expand that into further resolution. Some of the testers found this confusing when compared to the "Code Readiness" attribute. Maturity is effectively a high-level definition and qualifier to match the numbers – its resolving characteristics versus some of the other attributes is limited.

Verification: This attribute was taken directly from the PCMM table with a mapping into the TRL levels as shown above. There is extensive documentation in the SNL ASC V&V program that explains this attribute. This attribute answers two questions: (1) 'Are mathematical, algorithmic, and/or software errors degrading the readiness for application of the ModSim capability?' (2) 'Are numerical errors degrading the readiness for application of the ModSim capability?'

Validation: This attribute was taken directly from the PCMM table with a mapping into the TRL levels as shown above. There is extensive documentation in the SNL ASC V&V program that explains this attribute. This attribute answers the question: (1) 'Is the physical fidelity of the ModSim capability degrading its readiness for application?'

User Qualification: This attribute accounts for the level of expertise of the person using the tool for the problem that defines the TRL assignment context (see the "TRL Assignment Process" discussion below). It is intended to be applied taking into consideration the person doing the work, or the expertise of the person expected to do the work. If the analysis team was not already established at the time of the TRL assignment, it is expected that the manager of the capability area (e.g., thermal analysis) would make this TRL assignment.

Code Readiness: Also taken from the PCMM table, this attribute indicates the state in terms of code management and testing practices. The concept points more at the readiness for use by a user community (for example, can a user simply pick up the code and run it, or is there some probability that the code will not function properly) than a general statement of appropriateness for a given application, the latter being the entire point of a TRL. Readiness has to do with configuration management, stability of available software versions, availability of documentation,

support by code developers, and availability of appropriate computing hardware to at least execute calculations of interest, and so on. None of this implies that the code is "ready" for some particular application. Therefore, as used here, code readiness is just one part of a ModSim TRL assessment.

Models: This attribute was taken directly from the PCMM table with a mapping into the TRL levels as shown above. This attribute basically addresses the question 'What physics are important to the application and how physics-based are the models?'

Geometry: This attribute was adapted from the PCMM table with a mapping into the TRL levels as shown above. An important question that this attribute addresses is 'Are you overlooking important feature details that could significantly impact the results?' Our field tests indicated that the dimensionality of the problem was of less concern than the geometric fidelity – i.e., in some cases a high-fidelity, low-dimensional geometric representation was completely sufficient for obtaining the required analysis result. The most recent field tests provided further feedback indicating that in some cases the required analysis results can be obtained using medium geometric fidelity models, but this should not result in an overall lower TRL. We have yet to resolve this issue in the table, nor do TRLs in and of themselves bring clarity to this issue.

QMU: This attribute was taken directly from the PCMM table with a mapping into the TRL levels as shown above. It refers to elements of ModSim readiness that are pertinent to QMU, and is discussed in the Pilch, Trucano, and Helton report referenced above.

System: This is an indicator of the TRL for the computer hardware system used to do the computations. It is expected that the TRL assignment on that system is done 'externally' – i.e., done separately from an analysis-based capability TRL assignment. 'System' includes what may also more generally be called 'infrastructure' enabling ModSim, including storage and communication systems. Evaluation of the 'readiness' of complex hardware architectures for ASC-scale ModSim is obviously nontrivial and well beyond the scope of our initial TRL assessment effort.

Unfortunately, the columns that were taken directly from the PCMM table were not used extensively in the field tests. Some of the example problems did not use V&V or QMU at all, and in other cases the users were confused by the definitions. To resolve this issue, we either need to adapt the PCMM to clarify the wording in the columns, or list a point of contact that can answer such questions. As noted in the executive summary above and recommendations below,

we recommend performing the TRL assignments as an adjunct to the PCMM process, and that should directly address the clarity issue.

Tables 1 and 2 are intended to be used in the TRL assignment process as follows:

1. For each applicable column in the table, choose the definition that most accurately describes the node in question. For example, if you are using a relatively new physics-based model, you would probably choose the box that spans levels 6 and 7 in the Model column.
2. If the result of step 1 spans multiple rows, use your discretion as to which row it should be assigned. For instance, if the example in step 1 was very reliable and had a number of users, you would probably choose level 7 over level 6.
3. After you have done steps 1 and 2 for each column, the TRL is then selected as the minimum of those levels. For example, if Model was a 3, Geometry was a 9, and the rest of the levels were 6, the TRL would be a 3.

TRL Assignment Process – Proposed Methodology

The next step is to describe how to use the TRL definitions to produce a readiness level for the required capability. We have constructed a process methodology for this purpose, which is depicted in Figure 1 below.

On the left side of the figure are the ModSim TRL definitions and their direct mappings to NASA, DoD, and NWSMU TRLs, as described in the previous section. These mappings are important because they show that ModSim capability readiness levels can be compared to those of weapon hardware, etc., if need be.

At the top of the diagram is the capability being evaluated and its problem context. The capability can be something that was worked on in the past or something new that is planned for the future. It is usually given as the starting point of the analysis, although it might be necessary to clarify the details related to context. This problem context is important because it forms the basis for the remaining steps in the process. For example, the same feature in a software code could have a very different TRL depending on what is being done with it.

After defining the capability and problem context, the next step is to create a dependency tree that identifies the components needed to perform the capability. The dependency tree is exactly what its name implies, a way to represent dependencies between high and low-level capabilities (it is

not, however, meant to represent priority or parent-child relationships, a question that was frequently asked in the field trials). The capability and problem context form the top node in the tree, and the rest of the nodes are filled in beneath it. Each of these nodes is a capability in itself (or a sub-capability of the node above it, depending on how one looks at it). Sub-capabilities can take on any shape or form, provided that they sufficiently describe the software, hardware, and expertise required by their parent capability.

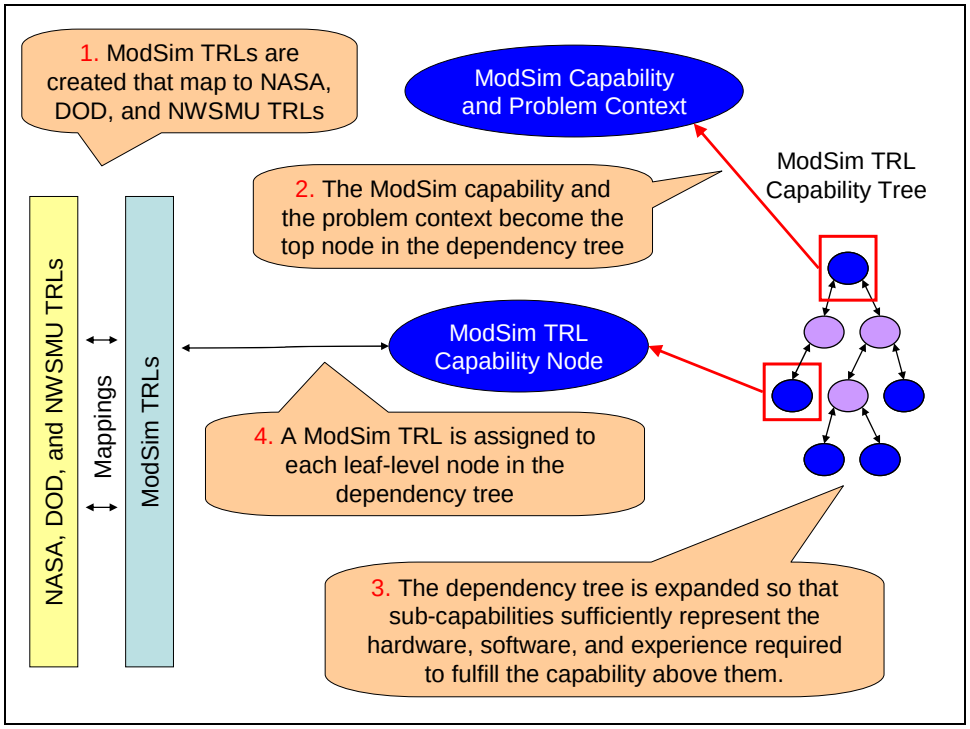

Figure 1 – ModSim TRL Assignment Process

Since there are multiple ways to view the same problem, the tree can take on a number of shapes and forms. However, all of the trees in our analyst examples (explained in the next section) looked remarkably similar, suggesting that a generic dependency tree template could be used as a starting point to save time in future evaluations. On the other hand, a DART example developed by Sean Brooks (Appendix 6) produced a unique tree compared to the others, which suggests that we might have different templates for different classes of problems. Both of the dependency tree templates that were created and approved by the analysts are included in Appendix 6. The notion of termination criteria, or at what point the tree expansion stops, will be addressed in the next section.

Once the dependency tree is created, the last step is to assign readiness levels to the leaf-level (i.e., bottom) nodes using the TRL definitions. It is first necessary to decide which columns in the table apply to each node, since some are only targeted towards software, hardware, etc. Next, a TRL is assigned to each of those columns, recording notes for each. These notes put justifications behind the numbers, and could eventually get captured in a final report. We found that a matrix was an easy way to record this information, where the rows are the names of the leaf-level nodes and the columns are the same as the definitions table. Then a number is marked in each box that applies, recording notes for the rows as the matrix is completed. Example matrices can be found in Appendix 5.

We initially envisioned an aggregation process, where leaf-level TRLs would percolate up the tree according to some aggregation calculus, and eventually yield a TRL for the entire capability. However, aggregation was a hotly debated topic with the steering committee, so we decided to focus our efforts on the TRLs of leaf-level nodes, since they would be necessary whether we ended up aggregating or not. One of the analysts suggested that we assign rankings to leaf-level nodes while computing their TRLs, and to use those as weights during the aggregation process. His reasoning was that certain nodes are more important than others, and it is hard to describe those relationships ahead of time without first constructing the tree. Of course, this mainly shifts conceptual difficulties to the challenge of quantifying 'weights' rather than eliminates them. Whether we end up aggregating TRL information, or simply looking at the leaf-level TRLs, is a decision that is still being debated within the WSC program at the time of writing.

Analysts Examples and Use Case

In this section we will describe the analyst use case, the full specification of which can be found in Appendix 1. We initially created this use case to get an idea of how ModSim TRLs would actually be used in practice within the Sandia ModSim community. However, it also gave us an opportunity to test the key assertions, TRL definitions, and TRL assignment process described in previous sections. We targeted analysts as opposed to developers and engineers because of their personal familiarity with the wide range of capabilities that are necessary to solve ModSim-related problems. The analysts that helped co-author the use case are Jay Dike (SNL/CA, multiphysics/mechanical analysis), Jeff Gruda (SNL/NM, mechanical analysis), and Roy Hogan, Jr. (SNL/NM, thermal analysis). We also met with Sean Brooks (an expert geometry and meshing model builder) at a later point, who gave us feedback from the Design Through Analysis Realization Team (DART) perspective. As can be seen, the analysts work in different locations

on different classes of problems, which helps to make the use case more representative of Sandia's work as a whole. However, it is also clear that yet more work could be (should be) performed to develop a more systematic experience base.

We went through the same set of steps with each of the analysts. First we discussed the overall process so that they understood what we were doing and why we were doing it. Then we helped come up with a specific problem and problem context. We had them pick something they had worked on in the past, so that they were familiar with the software tools, hardware, and expertise required to solve the problem. We also wanted the problem to cover as much of the analysis process as possible, from geometry and mesh creation to post-processing and visualization of ModSim information. Once that was complete, the analysts created a corresponding dependency tree, and then used the TRL definitions to assign readiness levels to the leaf-level nodes in the tree. For this step, we created a matrix where the columns were borrowed from the TRL definition table, and the rows were the leaf-level nodes (example matrices can be found in Appendix 5). For each row, we ran through the columns, and assigned TRL numbers to each that applied. We also captured notes for each node, which would document the justification if this were to be packaged into a final report. Finally, we captured analyst's thoughts on the complete process and its utility to them.

A question that came up in all of the examples was that of termination criteria, or the point at which tree expansion stops. The analysts showed that tree expansion could go on forever, unless one made a conscious decision to halt at some point. Our advice as facilitators was to do this at whatever place made the most sense to them. Initially we thought this might depend on things like accuracy and maturity or alternative capabilities. However, we soon realized that it was a function of TRLs. *New nodes should be created only when something requires further explanation and a TRL cannot be accurately assigned to the current node.*

While we did not ask the analysts to try to assign aggregate TRLs, the topic came up often during the exercise. Roy noted that all of the leaf capabilities can be in good shape, but integration is where the most problems surface. Because of this, he viewed aggregation as an important part of the process, as do we. Jay took this a step further and suggested that we assign rankings to the leaf-level nodes along with the TRLs that would essentially turn into weights during the aggregation process.

The analysts liked the column format of the TRL definitions because it let them apply filters to different types of nodes. For instance, it is possible to have one node for a software code and

another for a piece of hardware, have different columns apply to each, and yet still arrive at an equivalent TRL definition. They were also concerned about the effect of certain columns such as "Geometry" because in some cases they only needed "medium fidelity" models (which are a TRL of 5 or 6) to match the test data. However, in our current model, those would percolate through and lower the overall TRL, so we need to rework the wording in the definitions to prevent this from happening. The analysts were also confused on the "User Qualification" column because they were not sure if it depended on who was doing the analysis or who could be doing the analysis. In the latter case, their argument was that there will almost always be someone at Sandia that is an expert with a given tool, so if we are evaluating how capable Sandia is to perform a capability, the "User Qualification" should always be a 9. However, the expert user will not always be available due to time constraints, so we must take into consideration who will actually be performing the analysis with the ModSim capabilities when assigning TRLs.

The analysts did not find much personal utility in the process (in other words, they were not particularly interested in assigning TRLs to ModSim capabilities). However, they did recognize their role in assigning TRLs, since they are familiar with the tools and are able to provide unbiased opinions (as compared to developers, who could be biased in regard to their particular tools). They were also able to provide answers in almost every step in the process, which would not be true for most of the other stakeholders. In all three cases, the dependency trees were created in under an hour, and TRLs were assigned to the leaf-level nodes in less than three hours of additional time, which includes writing notes to correspond with the numbers. Taken together, this means that the entire process could be completed in less than a day, which would probably decrease as the number of evaluated capabilities went up (the analysts mentioned that they tend to use the same methods over and over). Thus, the burden on analysts in this specification of the process seems to be relatively small.

Two of the analysts expressed serious concerns about how TRL assessment was going to be used by the WSC Program Office. They were worried that a low TRL would be interpreted as doing a bad job, and would result in lower funding. Or conversely, that a high TRL would be interpreted as having a mature code that did not need further substantial funding support. This is part of the reason why we created a separate use case specifically targeted at Program Office application of ModSim TRLs, as described in Appendix 7.

Program Office Use Case

In addition to the field trials and use case developed with the analysts, Paul Yarrington provided information for a program office use case. We refined this information into a more detailed specification. We have included this use case as Appendix 7 in this report. The use case addresses the areas of investment, communication, application, response, and planning. The reader is referred to the appendix for full details.

Conclusions and Recommendations

When we started this project, we knew that we were working on an interesting problem for a number of reasons. Our initial thinking indicated that while people had evaluated the readiness of software systems in general, *they had not applied TRLs to CSE modeling and simulation specifically*. The predominant use of TRLs has been in system hardware applications (and for certain kinds of software systems like avionics software), but in fact very little information is available about the application of TRLs to CSE software in general. A lot was learned from this exercise, and we have reached four important conclusions that are associated with our primary recommendations.

Conclusion 1: We can assign TRLs to ASC ModSim capabilities.

While we recognize that there is work remaining to refine the modeling and simulation framework and process, we have developed a baseline ModSim TRL framework and solution to the assignment process, as documented in this report.

Recommendation 1: Use the framework and process proposed in this report as a baseline for refinement.

Conclusion 2: Modeling and simulation TRLs and the PCMM process are connected.

As shown in Tables 1 and 2 above, we have defined the evaluation criteria for modeling and simulation TRLs to substantially overlap with those of the PCMM table. This is a result of a combination of factors, including: a) the inherent nature of understanding the usability state of a ModSim capability, and b) our aim to keep the two representations synchronized. It is important to acknowledge that while the TRL and PCMM tables share some criteria, they are not attempting to solve the same problem. The PCMM approach is primarily concerned with risk identification

and mitigation, while the TRL table is primarily concerned with estimation of readiness levels – those are not the same thing, although they share much in common.

Recommendation 2: Conduct TRL assessments as an adjunct to the PCMM process, not as a stand-alone exercise.

Considering the degree of overlap in the core criteria being measured, and the fact that those overlapping criteria are by definition components of a PCMM evaluation, we recommend that TRL assessments be included as needed as an augmentation to PCMM evaluations. Conducting a 'stand-alone' TRL evaluation would require one to assess many of the PCMM criteria in the process, and that should be done according to the formal procedures and guidelines specified by the PCMM team, not in a reduced or simplified form just to complete a TRL evaluation.

Conclusion 3: The modeling and simulation TRL framework and process are not complete – there is still work to be done.

This report documents the baseline framework and process for assigning TRLs to modeling and simulation capabilities. Four field trials confirm that the basic approach described is sound, although incomplete and unrefined.

While we have made progress, there are still things that need to be finished. Perhaps most important are the descriptions in the ModSim TRL definitions table. Many of the field test users were either confused by the original wording in the columns, or had suggestions for improvement. We have recorded their feedback in Appendix 1 and updated the tables, but further refinements are still needed.

As mentioned above, we ultimately need to resolve the aggregation issue. Modeling and simulation capabilities are naturally aggregated to solve classes of problems – i.e., tools are assembled into higher-level problem solving capabilities.

As for the bigger picture, one might conceive of evolving this framework into a corporate business practice, which would standardize ModSim assessment throughout SNL. More tactically, it would be useful to have some sort of web application that would make it easy (or at least easier) to create and modify dependency trees, assign TRLs to nodes in those trees, and track and search through capabilities that were evaluated in the past.

Recommendation 3: If the sponsors decide to proceed with this line of development, continue to refine the framework and process using the baseline approach described in this report.

Conclusion 4: There remain questions about the utility of ModSim TRLs.

One of our early discoveries was that even within our own small team there were substantially differing views of the utility of using TRLs for modeling and simulation capabilities. While some team members viewed TRLs as a helpful aide in representing and communicating our ModSim capability readiness levels, others were more skeptical about their application. This latter view was due in large part to the implications of oversimplifying the assessment of the maturity and readiness of these capabilities and the potential misuse of the information. Further, we noted a range of opinions in our initial field tests with the analysts, from those that considered it useful to those that were wary of the misapplication of results (specifically, some analysts were concerned that a well-rated TRL might cause a code group to lose funding due to the implication that the code was sufficiently mature already).

In addition to the 'cultural' concerns above, there were fundamental concerns about the applicability of TRLs to advanced ModSim capabilities in general, since these capabilities are so substantially different in nature from the hardware that are usually associated with TRLs. Application of TRLs to ASC ModSim is not an obvious extension of the standard TRL usage paradigm. Substantial differences exist, including the following:

- There is no physical specification. The context for judging the readiness of a physical product (hardware) can be expressed as a physical specification (size, weight, performance measures) applied in a context (e.g., F-15 instrument panel, specified G-force range, specified temperature range, etc). Most of the software components for modeling and simulation do not have such a well-defined specification and context – indeed, the problems being addressed at Sandia National Labs are often 'one off'.

- Application of modeling and simulation capabilities is unique. Whereas the hardware being produced is typically being made in some quantity greater than one, modeling and simulation results are virtually always unique to the problem context.

- Application of modeling and simulation capabilities is not static. Whereas the hardware being produced is made according to an essentially static specification and application context, the exact opposite is true for modeling and simulation capabilities. The problem

context is entirely dependent on the analysis objectives, and the underlying tools (e.g., simulation codes) are continuously changing. In some cases, the problem solution requires problem-specific extensions to the codes. Hence, the readiness level of a modeling and simulation capability is in general time and problem context dependent.

So, in addition to the utility concerns above, there remain differences of opinion about the appropriateness of applying TRLs to advanced modeling and simulation capabilities.

Another debate about TRLs for ModSim focused on aggregation. We all agreed that we needed to break ModSim capabilities into pieces (which are represented as leaf-level nodes in the capability-dependency tree), but we could not reach a consensus on how to aggregate the information to arrive at a final TRL for a high-level capability. We originally proposed aggregating via some predefined calculus, such as an averaging method or the weakest-link-in-the-chain (minimum of all relevant TRLs contributing to a given capability assessment). However, not everyone agreed that it was possible to define an aggregation scheme ahead of time, and they were also worried about the same loss of information issue discussed above. Because of this, we decided to put aggregation on hold, but at some point we will need to revisit the issue. As currently defined, WSC use of TRLs requires an aggregation procedure.

Recommendation 4: Address key issues before proceeding to the next phase of developing the TRL framework and process.

Acknowledgements

The authors are grateful to the steering committee members, who worked through the problem with us to provide both guidance and direct help with the issues and problems. Specifically, we wish to thank Paul Yarrington as our primary sponsor within the steering committee, as well as Fran Current, Pete Wilson, Mike Hardwick, Scott Klenke, and Mike Chiesa. Also, we appreciate the guidance provided by our executive sponsor Art Hale.

We are also grateful to Jay Dike, Jeff Gruda, Roy Hogan, Jr. and Sean Brooks for their time and valuable help provided during the field trials. Their patience and insight was greatly appreciated.

Finally, we need to acknowledge Paul Yarrington for providing raw information from which we constructed the Program Office Use Case in Appendix 7 in this report.

Glossary and Acronyms

ASC – Advanced Simulation and Computing.

CSE – Computational Science and Engineering.

DOE – Department of Energy.

DP – Defense Programs.

ModSim – Modeling and Simulation.

NNSA – National Nuclear Security Administration.

QMU – Quantification of Margins and Uncertainties.

PCMM – Predictive Capability Maturity Model.

NWSMU – Nuclear Weapons Strategic Management Unit.

DSW – Directed Stockpile Work.

NW – Nuclear Weapons.

TRL – Technology Readiness Level.

V&V – Verification and Validation.

WSC – Weapon Simulation and Computing.

Appendix 1: Analyst Use Case

Modeling and Simulation Technology Readiness Level Analyst Use Case

Analysts
Jay Dike, Jeff Gruda and Roy Hogan, Jr.

ModSim TRL Team
Robert Clay, Scot Marburger and Max Shneider
Sandia National Laboratories

3/26/2008

Background

This document describes a generalized analyst use case for modeling and simulation (ModSim) technology readiness levels (TRL). It was written as part of a study examining the use and utility of ModSim TRLs for Sandia National Laboratories, from the analysts' point of view. This use case is one deliverable of a WSC program office effort to establish a framework to use TRLs as a means (in addition to other ongoing efforts such as V&V and PCMM) of describing and communicating readiness level of the ModSim capabilities the program provides in support of the NWSMU efforts.

The analysts and co-authors of this use case were:

- *Jay Dike* – SNL/CA multiphysics/mechanical analyst (8774)
- *Jeff Gruda* – SNL/NM mechanical analyst (1524)
- *Roy Hogan, Jr.* – SNL/NM thermal analyst (1516).

TRL Assignment Process

The discussion and results of the TRL assignment process as applied to the three analyst examples are presented in *Appendix 2*. In this section we will review the high-level process and comment on the roll the analysts played in generating the dependency trees presented in *Appendix 4* and the TRL assignment matrices presented in *Appendix 5*.

Figure 1 (within the SAND report) describes the high-level process flow for TRL assignments. This basic process was followed by the analysts developing the examples for this use case. A summary of the key steps follows:

1. Define the problem context for the capability TRL assignment.
2. Generate the capability dependency tree.
3. For each "leaf" in the tree, assign a TRL and document reasoning.
4. Validate the TRL assignments.

Step 1 was a simple matter – each analyst selected an analysis problem they were either currently. Their selections were as follows:

- *Jay Dike* – Abnormal/mechanical tension test of 304L using EMMI model, LS-Dyna3D, to failure.
- *Jeff Gruda* – Abnormal/mechanical penetrator, new fuse, new design LDRD.
- *Roy Hogan, Jr.* – Abnormal/thermal W80 V&V ModSim Milestone for WES Mock-2 with experimental data.

Step 2 proved easier than anticipated, since each tree was generated in an hour or less. Despite the fact that the tree depths were relatively consistent, some unexpected complexities arose. All three analysts asked questions about termination criteria, or the point at which you stop expanding the tree. Initially, there were many different theories as to when you should stop creating sub-nodes (discussed in *Appendix 2*), but we ultimately found that you should halt at places where you can naturally assign TRLs. Creating sub-nodes beneath those spots would only result in extra work, since it would be hard to assign TRLs to them.

The analysts weren't sure which way the arrows pointed between nodes (they thought that in some cases there should be directed edges from parent nodes to children, and in other cases the edges should be bi-directional). The main reason this came into question was because they wanted certain parts of their trees to be iterative (to represent, for instance, optimization loops). However, the analysts were trying to use the trees for more than their intended purpose, to show dependencies. It is up to the analysts to define the process flow that uses those dependencies. The three analyst dependency trees can be found in *Appendix 4*. As you can see, the trees all have similar structures, which seemed to suggest that a template tree could be used as a starting point to save time in future evaluations. This dependency tree template has been created and approved by the analysts, and is included in *Appendix 6*.

For step 3, we initially attempted to use a set of ModSim TRL definitions that were essentially a modest transformation of the NASA, DoD, and Sandia NWSMU 'hardware-based' TRL definitions. However, the analysts found the language vague and in general not an adequate representation of the key attributes associated with modeling and simulation capability readiness. Because of this, we constructed a new set of definitions (which can be found in *Tables 1 and 2*) that were better suited to the task and preferred by the analysts. They especially liked the new column format because it let them apply filters to different types of nodes, and yet still arrive at an equivalent TRL definition. The actual TRL assignments and corresponding notes for each example problem can be found in *Appendix 5*. While we did not ask them to assign aggregate TRLs, the topic came up

naturally during the exercise. All of the analysts thought that aggregation was important, but they agreed that coming up with a solution was non-trivial.

Discussion of Utility

The analysts did not find much personal utility in the process (in other words, they weren't particularly interested in what TRLs were assigned to ModSim capabilities). However, they did recognize their importance in the TRL assignment process, since they're familiar with the tools and are able to provide unbiased opinions (as compared to developers, who are most likely tied to their particular tools). They were also able to provide answers in almost every step in the process, which would not be true for most of the other stakeholders. And perhaps most importantly, they agreed that it wouldn't be too much of a burden to evaluate the readiness of ModSim capabilities from time to time, if asked to do so.

Two of the analysts expressed serious concerns on how this was going to be used by the Program Office. They were worried that a low TRL would be interpreted as doing a bad job, and would result in lower funding. This is the part of the reason why we created a separate use case specifically targeted at the program office.

A more detailed discussion of utility can be found *Appendix 3*.

Appendix 2: TRL Assignment Process Details

A high-level process flow has been defined that describes the necessary steps for the evaluation of a particular capability, which can be summarized as follows:

> A team consisting of at least analysts and possibly engineers, developers, and other constituents:
> 1. Creates a capability-dependency tree of supporting technologies beneath the requested capability, as defined by the problem context,
> 2. Assigns and documents ModSim TRLs to each leaf node in the tree,
> 3. Optionally, aggregates the leaf-level TRLs up the tree to get an overall TRL.

In step 1, the capability request essentially becomes the top-level node, and a dependency tree is expanded beneath it. The leaf-level nodes in this tree represent the software codes, hardware, and expertise needed to solve the problem, in context, from beginning to end, while the edges represent dependencies between nodes. In step 2, each leaf-level node is assigned a ModSim TRL based on the level definitions. As described in step 3, these leaf-level TRLs might also be aggregated up the tree to produce an overall TRL for the requested capability in the specified problem context (i.e., as defined in the top node of the tree). There are various aggregation schemes that we could apply, such as taking the lowest TRL of the leaves, or simply averaging the leaf TRLs. However, aggregation has been a source of contention among the steering committee, so for the time being the focus will be on developing the level definitions and methodology for assigning TRLs to the leaf nodes, tackling the aggregation problem after new knowledge has been gained in the process.

It's worth noting that this is a dynamic process, and a new tree must be created and expanded with each problem invocation. This is because a TRL doesn't make sense without a specific problem context. PRESTO, for instance, might have entirely different readiness levels from one context to the next. Furthermore, a readiness evaluation is also a snapshot in time since the requirements and capability components (such as software codes and hardware) are constantly changing.

Several experienced analysts, representing an initial sample of the analyst community, were chosen to test the TRL assignment process and co-author this analyst use case:

- *Jay Dike* – SNL/CA multiphysics/mechanical analyst (8774)
- *Jeff Gruda* – SNL/NM mechanical analyst (1524)
- *Roy Hogan, Jr.* – SNL/NM thermal analyst (1516)

The analysts represent two different locations (New Mexico and California), and two different classes of problems (mechanical and thermal). Each of the analysts performed the following tasks as part of the exercise:

- Selection of a specific analysis problem (defining the top-level ModSim capability and TRL assignment context)
- Expansion of a dependency tree beneath that capability/context
- Assignment of TRLs to the leaf nodes in the capability dependency tree
- Documentation of those assignments
- Discussion of the process and its utility.

For step 1, we asked each analyst to select a specific modeling and simulation problem to use as the basis for their example. These examples had to be complex enough to produce a nontrivial dependency tree. They were based on the analysts' previous work, so that they'd be familiar with the individual components required to complete the problems. Their selections were as follows:

- *Jay Dike* – Abnormal/mechanical tension test of 304L using EMMI model, LS-Dyna3D, to failure.
- *Jeff Gruda* – Abnormal/mechanical penetrator, new fuse, new design LDRD.
- *Roy Hogan, Jr.* – Abnormal/thermal W80 V&V ModSim Milestone for WES Mock-2 with experimental data.

Each analyst created a dependency tree for step 2. Jay actually started off by creating a list of components, and then grouping the terms before creating the diagram. Jeff and Roy, on the other hand, simply went straight to the diagram, brainstorming as they went. In all three cases, the tree was created in under an hour, which was less time than anticipated.

One question that came up in all three meetings was that of termination criteria, or the point at which you stop expanding the tree. The analysts realized that you could basically expand the tree forever, unless you made a conscious decision to halt at some point. Jay viewed the termination criteria as a function of accuracy and maturity. If the tool or feature had risk and wasn't guaranteed to give perfect results, he made it into a new node. If, on the other hand, it always gave the correct answer, it simply factored into the TRL of its parent node. Jeff had an entirely different termination perspective based on choices. When he expanded the tree, he'd usually get down to a level with a list of options that could all be used to solve the same task. His termination condition was then picking an option from the list, which could depend on many factors. For instance, one material model might require fewer tests than another, or one might select a tool because there's an expert down the hall that can answer questions about it. As it turned out, TRLs were the deciding factor, and we recommended that they stop at places where you can naturally assign TRLs to nodes. Creating sub-nodes beneath those spots would only result in extra work, since it would be hard to assign TRLs to them.

When Jeff and Roy constructed their capability dependency trees they both questioned the directions of the arrows between nodes. We had simply assumed that the arrows were directed edges connecting parent nodes to their children, providing an overall downward flow from the requested capability to the leaf-level nodes. However, both Jeff and Roy agreed that in some cases the edge arrows were actually bi-directional. They explained that the node might give results, and depending on what they are you could either accept them and move on or tweak some parameters and try again. In effect, the tree would change as the analyst work the problem and adjust their decisions based on what they learned.

One of the reasons that directions of arrows came into question was because the analysts thought that the tree should be iterative. We imagined that you'd start at the top of the tree and incrementally work your way down, but it turns out that in some cases the tree might contain loops. For instance, Jeff's example contained an optimization branch, which meant that you'd do everything else in the tree a certain number of times, either by hand or using a tool like DAKOTA. Roy actually took this a step further and drew a loop at the top of his tree, which meant that you'd do everything in the example one or more times. However, the analysts were trying to use the trees for more than their intended purpose, to show dependencies. It is up to the analysts to define the process flow that uses those dependencies.

We initially attempted to use a set of ModSim TRL definitions that were essentially a modest transformation of the NASA, DoD, and Sandia NWSMU 'hardware-based' TRL definitions. However, the analysts found the language vague and in general not an adequate representation of the key attributes associated with modeling and simulation capability readiness. Because of this, we constructed a new set of definitions that were better suited to the task and preferred by the analysts. They liked the column format of the TRL definitions because it let them apply filters to different types of nodes. For instance, you could have one node for a software code and another for a piece of hardware, have different columns apply to each, yet still arrive at an equivalent TRL definition.

They were also concerned about the effect of certain columns such as "Geometry" because in some cases they only needed "medium fidelity" models (which are a TRL of 5 or 6) to match the test data. However, in our current model, those would percolate through and lower the overall TRL, so we need to rework the wording in the definitions to prevent this from happening. The analysts were also confused on the "User Qualification" column because they weren't sure if it depended on who was doing the analysis or who could be doing the analysis. In the latter case, their argument was that there will almost always be someone at Sandia that's an expert with a given tool, so if we're evaluating how capable Sandia is to perform a capability, the "User Qualification" should always be a 9. However, the expert user will not always be available due to time constraints, so we must take into consideration who will actually be performing the evaluation when assigning TRLs.

While we did not ask the analysts to try to assign aggregate TRLs, the topic came up naturally during the exercise. Roy noted that all of the leaf capabilities can be in good

shape, but integration is where you run into problems. To paraphrase Roy, you can have a box full of perfect hammers, but it doesn't make you a sculptor. For instance, you can assign a mesh, but there's no place in the current model that describes how it works with the other pieces. For this reason, he thought that aggregation was an important part of the process. Jay didn't address this issue directly, but he did think that aggregation was possible, although he wasn't sure which aggregation scheme to use.

Appendix 3: Detailed Discussion of Utility

Jeff and Roy were first asked about the value they found in the process, and both gave similar answers. Neither thought that it helped them much directly, and they couldn't see themselves doing this on their own unless their managers requested it. Roy explained that when analysts get a problem, they can either do it with the tools they have at hand or they can't. Ratings don't really help them solve their problem. As project leads, they typically want an effect, not an approach. They ask their team members for the answers, and usually don't care what tools they use to get it. They did agree that this could be a means for analysts to shop around and find other tools to use, but unfortunately they thought analysts might only look for tools once a year, or a few times in their career.

However, both recognized the importance of analysts in the TRL process, since they know and work with the tools in question, and are able to provide unbiased opinions (in the sense that they aren't tied to one tool over another). They were also able to provide answers to almost every step in the TRL process, which would not be the case with most of the other stakeholders. And they all agreed that if they had to evaluate the readiness of a ModSim capability from time to time, it wasn't too much of a burden. In each example, the dependency trees were created in under an hour, and TRLs were assigned to the leaf-level nodes in under three hours (of additional time). This includes writing notes that correspond with the numbers, but it does not include the time that's necessary to wrap the notes in an overall report, if required. Taken together, this means that the entire process could be completed in a day, which would probably decrease as the number of evaluated capabilities went up (the analysts mentioned that they tend to use the same methods over and over).

All of the analysts agreed that the TRL assignment process was probably of more use to the WSC program office. They understood that it could be a communication mechanism, since TRLs are already used in other parts of Sandia as well as various government agencies. They also noted that TRLs could help guide investment decisions by and identifying capability weaknesses. However, Roy's main concern was that if a tool was assigned a 7 or an 8, it would cause the WSC office to declare victory. He wouldn't want to see a decrease in investment just because a project is doing well, since tools are always improving. Roy observed that the class of problems that they are solving is a moving target, and today's TRL 8 could be tomorrow's TRL 3 based on the changed problem context. These suggestions explain why we're also creating a separate WSC Program Office use case.

Appendix 4: Analyst Example Dependency Trees

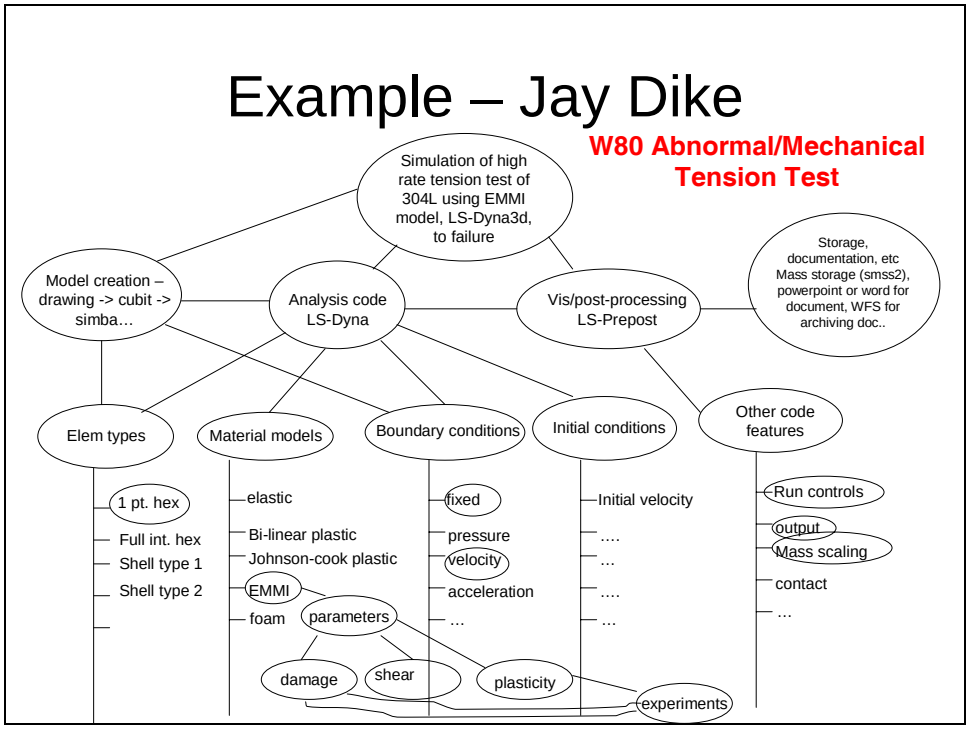

Figure 2 – Jay Dike's dependency tree

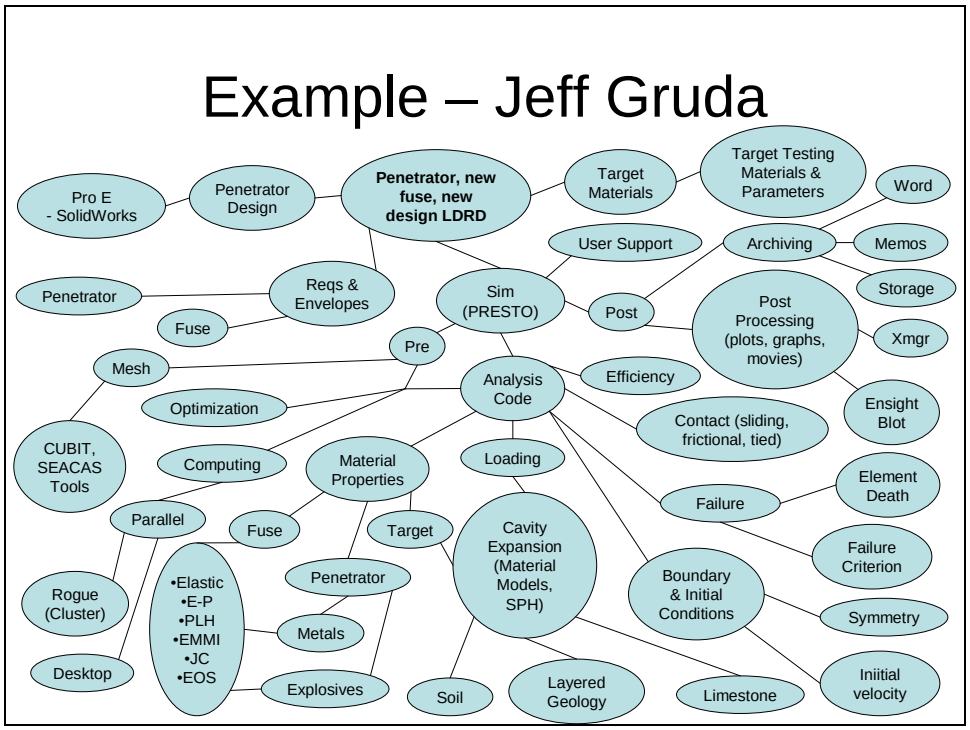

Figure 3 – Jeff Gruda's dependency tree

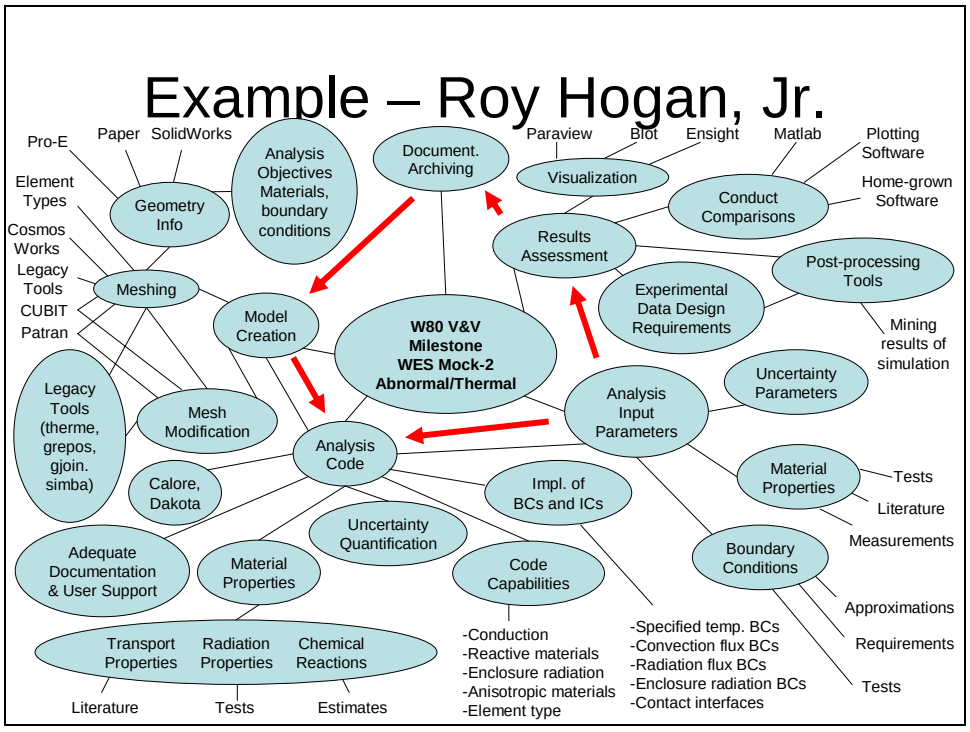

Figure 4 – Roy Hogan, Jr.'s dependency tree

Appendix 5: Analyst Example TRL Assignment Matrices

Jay Dike's TRL Assignments

1. This particular simulation was able to run on a desktop, so no cluster/parallel issues
2. Would use LS-Prepost here. Not a 9 on maturity because it would be nice if it was faster, and because of that, they don't do it as much as they'd like to. Also, in general, some of the features aren't there (doesn't have transparencies, can't represent all of the variables you want to in the output, can't get orientation/tracking easily). Also, can't do certain things in batch mode. 7 for code readiness because there are times where you get weird answers because sometimes versions between analysis code and post-processor don't match up. Sometimes they also have something implemented that's not quite right. Room for improvement with performance between remote and desktop.
3. Using LS-Prepost here as well, so similar to (2). LS-Prepost is not very good on remote platforms. No mining in this particular example.
4. A lot of times it doesn't work very well (talking about HPSS). It is production, which is why it's a 7 and not a 6. Some of the connections between platforms aren't reliable (for instance SMSS).
5. Includes WebFileshare and Sharepoint
6. They did do uncertainty, even though it wasn't on original diagram. There are codes and tools associated with this (Dakota, etc.) that would be easier to assign TRLs to. If we were doing more of a parameter study, this would be applicable and there would be more bubbles under this one that we'd assign more levels to.
7. 9 for this problem – they can do it, and they think it's done correctly. Other cases, they might go in and check. Fixed is special case of velocity.
8. EMMI as been applied to a number of realistic problems, but we know that there are a number of things it should doesn't do or should do differently. For verification, knows that he can quantify the errors, but the errors are still going to be larger than he likes. Many times can get "is it going to break, or not", but would like more information than that. There has been quantitative validation against test, but there's still a lot to sort out. For user qualification, a lot of people don't use it because they're not familiar with it. Jay's familiar, but not as much as the model guys, but in this case he has all of the expertise to solve this problem. Usability could be improved for selecting parameters for specific materials. Has some regression testing, but there's probably a lot more coverage that they could have. 7 for models because some forms of damage models that are still being sorted out (like shear). Experiments have a good readiness level (8, not a 9 because there are improvements we could make) – we know which ones we like to use. Getting the parameters for the EMMI model to match all 3 test results at once are more of a problem.

9. Each element type has inherit limitations, but that probably doesn't play into that
10. SIMBA doesn't know all of the features for any particular code. You just dump in the stuff that it knows about. Doesn't know all the run controls, so you just dump those into a text file that gets spit back out. So all in all, it does the most important things, but things that are easier for you to do manually, it leaves for you to do. Knows they do a lot of regression testing. Knows that it seems to work equally well on Linux and Windows machines.
11. Knows that they do a lot of regression testing and software quality.
12. Geometry info for this case is experimentalist just giving them a drawing, or at least dimension information

Node Name	Above Note	Capability Maturity	Verification	Validation	User Qualification	Code Readiness	Models	Geometry	QMU	System
Computing Hardware	1	9			9					9
Visualizations	2	8			9	7				7
Post-Processing Tools	3	8			9	8				7
Mass Storage	4	7								7
Documentation	5	9			9					9
Uncertainty Parameters	6									
Velocity BCs and ICs	7	9								
Fixed BCs and ICs	7	9								
EMMI Material Model	8	7	6	6	7	3	7			
Damage, Shear, and Plasticity Experiments	8	8								
Run Controls		9			9	9				
Mass Scaling		9			9	9				
1 pt. Hex	9	9			9	9				
Mesh Modification (SIMBA)	10	7			9	9				9
Meshing (CUBIT)	11	9			9	9				9
Geometry Information	12									

Table 3 – Jay Dike's TRL Assignment Matrix

Jeff Gruda's TRL Assignments

1. Doesn't really fit into any of the columns. Maybe user support should be its own column as well (manuals, FAQs, verification tests, ISO9000, phone support)
2. Never got to the point of optimization – he intended to use DAKOTA, but never got there. His user qualification was low, but that shouldn't drop down the TRL level. If he would have used DAKOTA, he would have called in a person to help.
3. Very vague because it was a design and the requirements moved a lot, which is the whole purpose of an LDRD. Used low geometry to find an optimum point and move on from there, but that shouldn't lower the TRL level. Pro-E and Solidworks do simple tasks well.
4. Meshing. 9 because CUBIT, SEACAS tools have been around and used. 3 because of geometry.
5. Code capabilities – code readiness probably a 7. 5 for verification – doesn't know how well contact models have been tested out – same for validation. Most of these are a matter of guessing, not of truth, since he's not necessarily the one who would be answering those questions.
6. Soil, layered geometry, and limestone should be under target instead of where they are now.
7. Cavity expansion – temper.
8. Only a couple of them, elastic, plastic.
9. Readiness and models: 7 – been around awhile, standard stuff. Not empirical.
10. PRESTO – 8 because it's been compared to other codes
11. Archiving – PowerPoint presentations. Wasn't a big thing that took 10 days that he had to run 5 times. Word and PowerPoint work pretty good.
12. Considers them at similar level as pre-processing tools – same types of tools, same usage.
13. Pretty rock-solid. Desktop and rogue both work pretty well.

Node Name	Above Note	Capability Maturity	Verification	Validation	User Qualification	Code Readiness	Models	Geometry	QMU	System
User Support	1									
Optimization	2									
Geometry	3	9								
Requirements and Envelopes	3							3		
Pre-Processing/Meshing/Element Types/Mesh Modification	4	9						3		
Code capabilities/contact	5	6	5	5		?	?	N/A	?	
Failure	6	5	5	3			5			
Loading	7	7	6	5			5			
Material Models	8	8					8			
Material Properties	8			4						
BCs and ICs	9	8				7	7			
Efficiency	10					8				
Archiving	11	7								
Post-Processing	12	9								
Computing Hardware	13	9								9

Table 4 – Jeff Gruda's TRL Assignment Matrix

Roy Hogan, Jr.'s TRL Assignments

1. Clusters for this problem – 8 because it's production, and compared to other cluster systems, it's probably in the middle. 8 in user qualification, because uses them a lot, but wouldn't say he's the number one expert at Sandia.
2. Using all 3, Paraview, Blot and Ensight. Focus on Ensight. Capability maturity – runs on most platforms that he's aware of, . Gives himself a 6 because he can do 7 or 8 because he can do most of what he needs to do, but does he know everything about Ensight? Definitely not. Explained to him that the latter was the way to go.
3. Matlab is what they mostly use on that. Basically has results for all post-processing tools, using Matlab to match them up because Matlab is very useful in doing that sort of thing. With some help (other team members), was able to get the job done. During this project, he learned how to use Matlab better, were other guys that were probably already an 8 or a 9. Code readiness, would guess that it would be pretty high. Also comparing with test data.
4. Post-processing tools really embodied by the Vis and Conduct Comparisons.
5. Dependency in the sense that you need the test data. For this project, maybe a 7 for validation because uncertainties in diagnostics, etc. Not a geometry of a numerical model, but geometry of a test. Some simplification of parts, but intended to have some representation of geometry. Test geometry almost always 3D.
6. Believed that some of this went into one of Marty's V&V databases. Marty would like it to be a standard, but it probably depends on your perspective. SAND report, corporate archiving. Particular runs – probably not. If you just store information to regenerate results, might not be able to in the future, since codes, operating systems, etc. change. Can't recover a previous generation of a code.
7. How well do you know the parameters that are associated with your problem. Based on what he calls validation – we believe that those tests covered the dominant physics. Geometry – basic geometry but test did have some block representation. 7 on approximations, requirements and tests based on QMU because we did sensitivity/uncertainty quantifications based on BCs. Specifically applied to what are the affects of the BCs.
8. 8 because they quantified the uncertainty using formal methods.
9. Material models – more mechanical – doesn't really apply to thermal
10. Some you know well, some you don't know as well. 5 or 6 under models because in some cases models were calibrated, and in some cases it was a 9. Radiation properties, might not know emittance as well. 4 under validation because uncertainty under properties is large unknown.
11. Could run this one up to talk about analysis code itself. Could split them out, but doesn't think we want to go to that level of detail. Figured out mathematical model – binary, even you put them in or you didn't. 8 under models, maybe a 7. Would not be an 8 or a 9 because some of the foam decomposition models are not first principle – require some parameters. About a 7 on code readiness because he

doesn't know the coverage or the high order interactions (coverage when exercised with multiple features). User qualification probably an 8.

12. On this project, they didn't use SIMBA or the SEACAS/legacy tools. So meshing/mesh modification could almost be put into 1. On this project, they used mostly Patran and Therme. On user qualification, if it's the guys in 2900, probably give them an 8 (they made the mesh as he recalls). If it's him, probably a 6 – maybe less on SIMBA when they use that (not on this project). Big distinction in user qualification between can you operate the tool well enough to get your job done, or are you an expert in it. Probably about a 6 on geometry – had to capture key aspects, but had some simplification (cheaper to build simplified version than version they gave us. Put in surrogate strong link that had same features, but not as much detail – to cast one of these bases, costs a lot more). Code readiness – pretty good because they're either commercial, or tools they've used for quite awhile.
13. Probably combination of Pro-E and Solidworks. In terms of code readiness, both of those are about an 8. Geometry – 5/6, getting repetitive because there are simplifications, since that's the level of detail they needed in the model. So geometry is always low because it reflects the test – didn't need a more detailed model. 7/8 on code readiness because Pro-E and SolidWorks are pretty solid. Probably a 2 for Roy in Pro-E, but for guys who built it, they were 8s because it was a very good model. So long ago, hard for him to remember these types of details.
14. Optimization is iteration in Roy's case. You're doing the entire problem multiple times – implied. Probably lots of parts of tree you're doing multiple times. Hard to assign a TRL to this node individually.
15. Lot of information there for Calore, but could be improved in terms of providing guidance and instruction.

Node Name	Above Note	Capability Maturity	Verification	Validation	User Qualification	Code Readiness	Models	Geometry	QMU	System
Computing Hardware	1	8			8					8
Visualization Tools	2	8			8					
Conduct Comparisons	3	8			7	8				
Post-Processing Tools	4									
Experimental Data Design Requirements	5			7				6		
Documentation, archiving	6	7/8								7/8
BCs and ICs	7	7						6	7	
Uncertainty Parameters	8		8						7	
Material Models	9									
Material properties	10			4			5/6			
Code capabilities	11				8		8			
Meshing/Mesh Modification	12									
Patran	12				8/6	7/8		6		
Therme	12				8/6	7/8		6		
Geometry	13	7/8			2	7/8		5/6		
Optimization/Iteration Doc and user support	14 15	7								

Table 5 – Roy Hogan, Jr.'s TRL Assignment Matrix

Appendix 6: Dependency Tree Templates

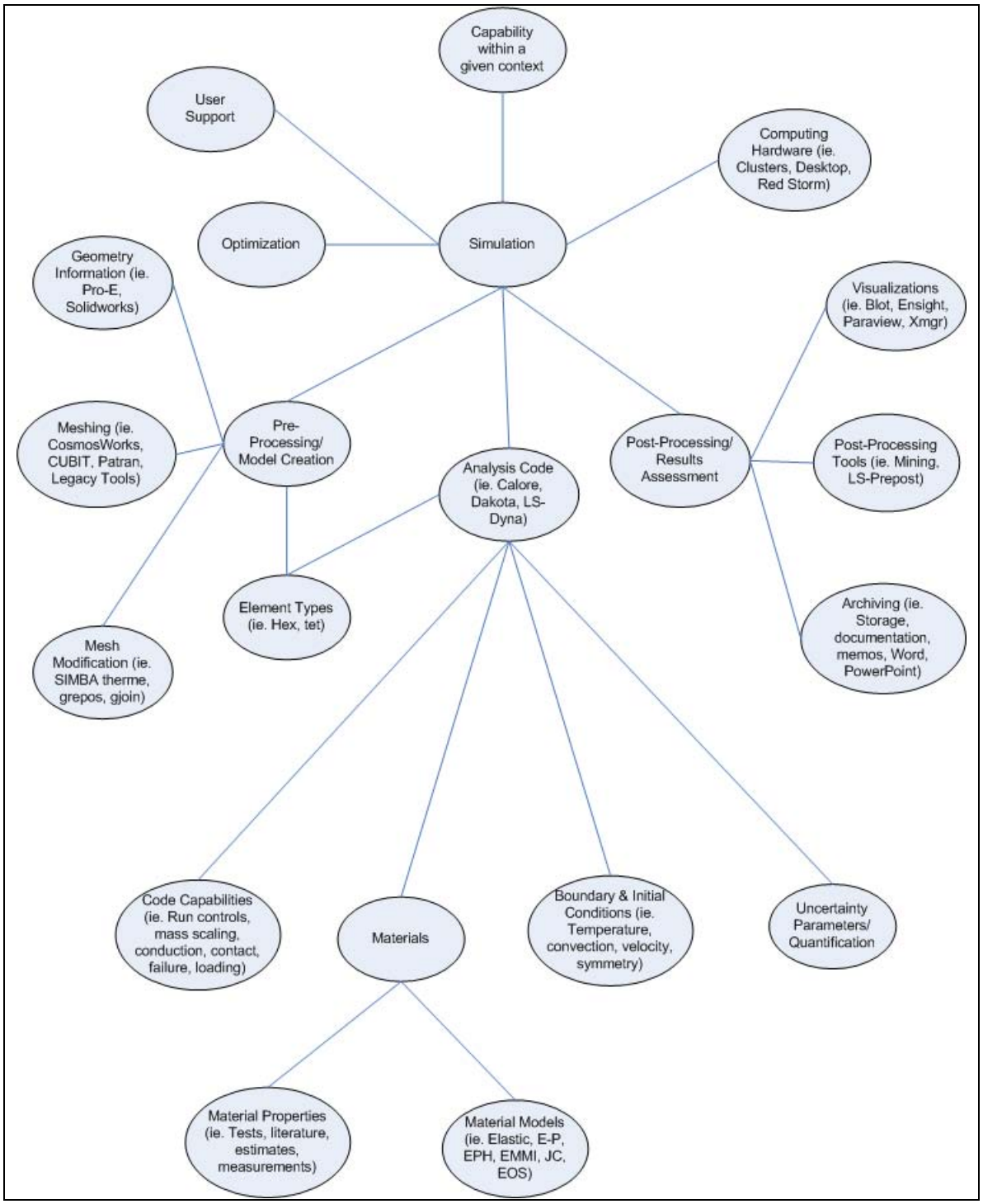

Figure 5 – Analyst Dependency Tree Template (Jay Dike, Roy Hogan Jr., and Jeff Gruda)

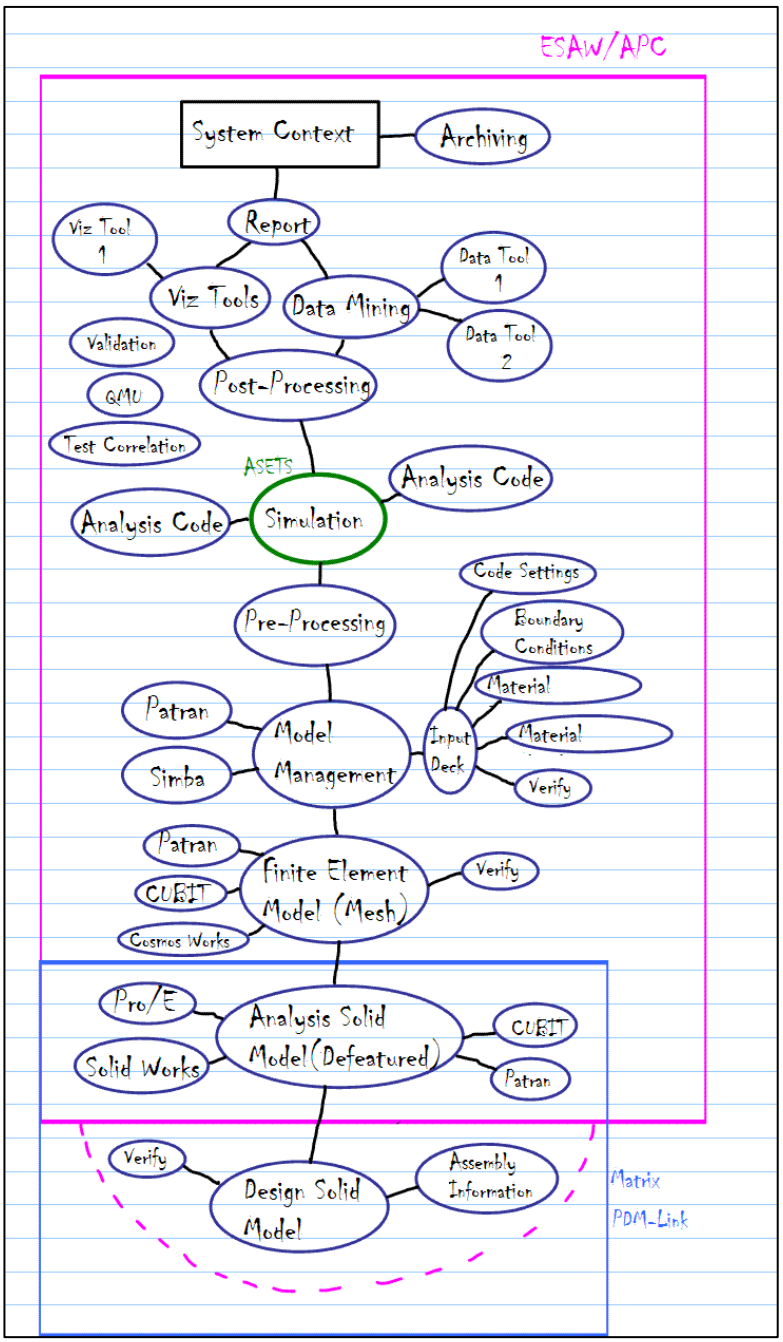

Figure 6 – DART Dependency Tree Template (Sean Brooks)

Appendix 7: Program Office Use Case

Paul Yarrington briefly summarized potential applications of TRL's for the WSC program office as follows in an email to our team. This information is summarized as follows:

- Investment: Look at representative applications in the various Focus Areas. Assess TRLs for corresponding compute capability required to support the representative apps. Decide if overall/aggregate TRLs are OK. If not, target investments to raise TRL of weakest underlying sub-capability to get TRL to desired threshold for representative apps.
- Investment: Provides a basis for decisions on buy versus build. Assume that can assess (roughly, at least) the TRL of some key commercial products. If in-house efforts will take "too long" to get to comparable TRL, then buy the capability and invest where SNL products are more competitive.
- Communication: Framework for common language to use in interactions and collaborations with external organizations. Facilitate decision on how to carve up responsibilities for advancing state of the art based in TRL of sub-capabilities from the various parties.
- Application: Provides a basis for assessing the uniformity of TRLs across a collection of sub-capabilities assembled to address some engng issues. Helps avoid overlooking a "weak link" in the sub-capability collection that produces an overall low TRL "product/tool/capability" with an otherwise apparently high TRL approach. (Clearly similar to the "response" use case above.)
- Response: Urgent request is received, say due to National emergency. Assemble M&S capability such that overall TRL is adequate. Identify any specific sub-capabilities that might be limiting overall TRL and replace with more mature alternative to get adequate confidence (TRL) in overall capability. Alternatively, could be viewed as providing rationale for resisting programmatic zeal to always use "latest/greatest" sub-capabilities that might (in principle and in the future) have more physics fidelity (and potential programmatic appeal) but are in fact lower in TRL due to immaturity of say the V&V.
- Roadmap: Identify the criteria and chart the expected timeframes for advancing through TRLs for given sub-capability/application. Gives metric for assessing progress... i.e. how much is enough and are we getting there fast enough?

We expanded upon Paul's initial description to create a 'use case' for the Program Office. This 'Use Case' targets 'Investment' but ends up encompassing 'Communication,' 'Response,' and 'Application.' 'Roadmap' remains slightly different. The presentation below is basically unmodified for purposes of this report, which preserves the 'look and feel'

Use Case Investment:

Inputs: (per Paul's general description)

1. Defined applications, A1 through AN. These are drawn from WSC Program Focus Areas.

    a. [Comment: Some focus areas have many applications associated with them. It is unlikely that N can be very large.]
2. Defined TRLs.
    a. [Comment: This is in progress.]
3. Defined M&S components for A1 through AN, say $C1^{A1}$, … , $CM1^{A1}$; $C1^{A2}$, …, $CM2^{A2}$; etc.
4. Assigned TRLs for corresponding component M&S capability needs for A1 through AN, call these TRL($C1^{A1}$), … , TRL($CM1^{A1}$); TRL($C1^{A2}$), …, TRL($CM2^{A2}$); etc.
5. Aggregated TRL for M&S capability for the application, written as TRL($C1^{A1}$, … , $CM1^{A1}$); TRL($C2^{A2}$, … , $CM2^{A2}$); etc
6. Specification of required/needed/desired TRL levels for application A1 through AN, call it LA1 through LAN.

Outputs: (per Paul's general description)
1. Decision: (a) If aggregate TRL is adequate (e.g. TRL($C1^{A1}$, … , $CM1^{A1}$) ≥ LA1, etc) , no supplemental funding needed [possible disinvestment? Funding required to maintain TRL? (b) If aggregate TRL is inadequate, identify components that weaken the aggregate and invest to improve their TRLs.

TRL needs implied by this usage:
1. A concrete definition of TRLs.
    a. [Comment: In progress, with some emphasis on CS&E software components.]
2. Systematic procedure for identifying M&S components required for specified applications for which TRLs must be evaluated.
    a. [Comment: Such a procedure has not been implemented currently, but note that this strongly correlates with the development of a PIRT [an ASC V&V program construct] for the intended application. The procedure can probably be characterized as a PIRT development task.]
    b. [Comment: There is an ongoing effort to define Dependency Trees that are useful for identifying separate M&S components for TRL evaluation. This is also linked to the Analyst Use Case development activity. We expect that this effort will be successful at application decomposition, but it does not directly address the problem below, that is, of specific TRL assignment. The current effort, however, suggests that an aggregation approach rooted in dependency trees may reduce to "lowest TRL in the dependency tree wins", at least for the investment use case. See below for more comments on aggregation.]
3. Systematic procedure for assigning TRLs to identified M&S components.
    a. [Comment: One analysis of this task is being developed in the current TRL investigation with the "Analyst Use Case" – that is, how analysts would evaluate M&S components from a given specification of TRLs. There is some expectation that this may be easier to accomplish at lower component levels, as well as some demand to do this (from Pete Wilson). But note that the lower the component level, the larger the actual number of component evaluations that have to be performed for each application.]

b. [Comment: (Marburger) – “We have started to standardize on terminology with respect to the various parts and elements of the TRL process. Some helpful definitions that are starting to emerge are:
    i. Capability - The collection of software, hardware, and expertise needed to deliver an analytical result in response to a customer request.
    ii. Component - one of the elements of software, hardware, or expertise needed to produce an analytical result in response to a customer request.
    iii. Context - the set of boundary conditions, expected results, funding, and schedule requirements negotiated by a customer and analyst that help to define a capability.
    iv. Dependency Tree - the list of components and their relationships to each other that comprise a capability, usually expressed graphically.]
c. [Comment: Note that multidimensionality is automatically appearing in the context of even terminology standardization. Multidimensionality requires collapse in the aggregation process. Thus, aggregation is not only combining separate TRL evaluations, but collapsing the inevitable multidimensionality.]

4. A TRL aggregation procedure.
    a. [Comment: This has not been defined at this point. The implication of the description provided by Paul is that the aggregate TRL is the minimum of the component TRLs. This idea has been discussed but not really analyzed.]
5. A means of specifying a required/needed/desired TRL level for the given application.
    a. [Comment: Specifying a required/needed/desired TRL requires involvement from the application side, if not outright ownership by the application side of this specification. Note that this level may naturally originate from the same analysis that provides a PIRT for decomposing the application, another advantage of thinking PIRTs.]
    b. [Comment: However the level is specified, involvement on the application side requires communication, thus Paul’s “communication” use case is subsumed under this use case.]
    c. [Comment: Communicating requires communicating the specified WSC TRLs to the application side, which may have other ideas about what TRLs mean. Thus, this communication probably puts constraints on how far the WSC language defining TRLs can deviate from the language of the application area. Remember the arguments that TRLs appropriate for advanced development M&S products are likely to be highly divergent from those appropriate to hardware products. Nonetheless, realistically some significant compatibility is likely a requirement. This is an essential tension that must be resolved.]
    d. [Comment: One approach is that WSC specifies required/needed/desired TRL level for given applications. This helps us move forward without having to solve the communication problem, but carries the same danger as inviting code developers to specify TRLs for their own software. We suggest that we need to deal with the communication use case as part of this use case.]

We believe that both the "response" and the "application" use cases mentioned by Paul are subsumed by the tasks that must be performed to achieve the "investment" use case. Thus, four of the five general use cases Paul proposes are subsumed by the "investment" use case. Emphasizing the "investment" use case provides broad value.

The "Roadmap" use case is a bit different.

1. Evaluating TRLs theoretically (but only theoretically) implies knowledge about how to elevate the TRL. This is harder the more the TRL is aggregate.
2. Achieving a higher TRL is a project planning exercise as well as an advanced development issue. Thus, built into the roadmap use case are factors like cost estimation, which has been a nontrivial problem for the ASC program in the past.
3. "How much is enough" is defined by a TRL that doesn't need to be exceeded. But remember that this is across a lot of components and involves an aggregation procedure.
4. Roadmap also implies dealing with uncertainty in TRL specifications. [Or do we really believe that these evaluations won't have uncertainty associated with them.]

Final questions based on the "investment" use case:

1. Will WSC identify A1 through AN?
    a. [Answer: (Yarrington) – SNL WSC and DSW should define these. ASC Focus Areas provide a reasonable structure for doing so. The NNSA HQ ASC "predictivity" performance indicator provides additional impetus for this. The current plan rests on selecting canonical applications for each Focus Area (then quantifying the performance level through some algebra applied to assessment based on PCMM taxonomy).]
2. How do we move forward on the aggregation procedure?
    a. [Comment: (Yarrington) – It is agreed that this is a difficult issue and the TRL team will have to provide progress on this.]
3. Confirm that the "Investment" use case M&S components include hardware (computers, systems, etc), software (Apps codes, system software, infrastructure, etc), and people (skill levels). Or define the restriction.
4. How will time dependence be handled? This has strong implications for the agility, expense and response time required/needed/desired of TRL evaluation procedures.
    a. [Answer: (Yarrington) – We are expecting the Focus Area structure to provide clarity around investment needs from one FY to the next, hence this becomes one way of at least enveloping time dependence. Accordingly, re-evaluation on an annual basis of TRL status for the Defined Applications for the various Focus Areas in preparation for budget and program plan decisions should provide a structure for the re-evaluation.]

Distribution

MS0104	Bickel, T. C.	(01200)
MS0139	Current, M. F.	(02998)
MS0139	Hale, A. L.	(01900)
MS0139	Thomas, R. K.	(01904)
MS0139	Yarrington, P.	(01910)
MS0321	Nelson, J. E.	(01430)
MS0370	Trucano, T. G.	(01411)
MS0372	Gruda, J. D.	(01524)
MS0374	Brooks, S.	(02991)
MS0376	Blacker, T. D.	(01421)
MS0380	Jung, J.	(01542)
MS0380	Morgan, H. S.	(01540)
MS0382	Sjaardema, G. D.	(01543)
MS0382	Stewart, J. R.	(01543)
MS0384	Ratzel, A. C.	(01500)
MS0427	Klenke, S. E.	(02118)
MS0469	Corbett, D. W.	(02900)
MS0469	Verardo, A. E.	(02990)
MS0631	Witek, H. M.	(02910)
MS0639	Terhune, G. M.	(02950)
MS0735	Merson, J. A.	(06310)
MS0801	Leland, R. W.	(04300)
MS0801	White, D. R.	(04340)
MS0821	Thornton, A. L.	(01530)
MS0824	Hermina, W. L.	(01510)
MS0836	Hogan Jr., R. E.	(01516)
MS0847	Baca, T. J.	(01523)
MS0847	Wilson, P. J.	(01520)
MS1002	Roehrig, S. C.	(06300)
MS1005	Skocypec, R. D.	(06340)
MS1104	Robinett III, R. D.	(06330)
MS1138	Mitchiner, J. L.	(06320)
MS1318	Womble, D. E.	(01410)
MS1319	Ang, J. A.	(01420)
MS1322	Dosanjh, S. S.	(01400)
MS9004	Damkroger, B. K.	(08130)
MS9004	Falcone, P. K.	(08110)
MS9004	Hruby, J. M.	(08100)
MS9004	Lindner, D. L.	(08120)
MS9042	Chiesa, M. L.	(08774)
MS9042	Dike, J. J.	(08774)
MS9042	Kistler, B. L.	(08774)
MS9104	Ortega, A. R.	(08229)
MS9151	Hirano, H. H.	(08960)
MS9151	Napolitano, L. M.	(08900)
MS9151	Oien, C. T.	(08940)
MS9152	Marburger, S. J.	(02998)
MS9153	Handrock, J. L.	(08810)
MS9155	Nielan, P. E.	(08116)
MS9159	Ammerlahn, H. R.	(08962)
MS9159	Clay, R. L. (5)	(08964)
MS9159	Hardwick, M. F. (5)	(08964)
MS9159	Shneider, M. S.	(08964)
MS9161	Pontau, A. E.	(08750)
MS9404	Kwon, D. M.	(08770)
MS9405	Carling, R. W.	(08700)
MS9409	Moen, C. D.	(08757)
MS9018	Central Tech. Files	(08944)
MS0899	Technical Library	(04536)