Software Technology Readiness for the Smart Grid

Cristina Tugurlan, Harold Kirkham, David Chassin
Pacific Northwest National Laboratory
Advanced Power and Energy Systems
Richland, WA

cristina.tugurlan@pnnl.gov, harold.kirkham@pnnl.gov, david.chassin@pnnl.gov

Abstract

Budget and schedule overruns in product development due to the use of immature technologies constitute an important matter for program managers. Moreover, unexpected lack of technology maturity is also a problem for buyers. Both managers and buyers would benefit from an unbiased measure of technology maturity. This paper presents the use of a software maturity metric called Technology Readiness Level (TRL), in the milieu of the smart grid. (The smart grid adds increasing levels of communication and control to the electricity grid.) For most of the time they have been in existence, power utilities have been protected monopolies, guaranteed a return on investment on anything they could justify adding to the rate base. Such a situation did not encourage innovation, and instead led to widespread risk-avoidance behavior in many utilities. The situation changed at the end of the last century, with a series of regulatory measures, beginning with the Public Utility Regulatory Policy Act of 1978. However, some bad experiences have actually served to strengthen the resistance to innovation by some utilities. Some aspects of the smart grid, such as the addition of computer-based control to the power system, face an uphill battle. Consequently, the addition of TRLs to the decision-making process for smart grid power-system projects might lead to an environment of more confident adoption.

Biography

Cristina Tugurlan *joined Pacific Northwest National Laboratory as a software test engineer in August 2010. Before joining PNNL, she held a software engineer position at IBM, OR. She is responsible for the software testing and validation of projects modeling wind generation integration, power system operation and smart grid. She has a Ph.D. in Applied Mathematics from Louisiana State University, Baton Rouge, LA.*

Harold Kirkham *received the PhD degree from Drexel University, Philadelphia, PA, in 1973 and joined American Electric Power, and was responsible for the instrumentation at the Ultra-High Voltage research station. He was at the Jet Propulsion Laboratory (JPL), Pasadena, CA, from 1979 until 2009, in a variety of positions. In 2009 he joined the Pacific Northwest National Laboratory, where he is now engaged in research on power systems. His research interests include both power and measurements.*

David Chassin *has been at Pacific Northwest National Laboratory for 19 years and has more than 25 years of experience in the research and development of computer applications software for the architecture, engineering and construction industry. His research focuses on non-linear system dynamics, high-performance simulation and modeling of energy systems, controls, and diagnostics. He is the principle investigator and project manager of DOE's SmartGrid simulation environment, called GridLAB-D and was the architect of the Olympic Peninsula SmartGrid Demonstration's real-time pricing system.*

1. Overview of Paper

The paper begins by explaining how regulation of the power utilities led to a lag in the implementation of distributed automatic control. Re-regulation has changed the situation. The paper points out the need for assurance that any control now being introduced is ready for application.

The second section of the paper describes how Technology Readiness Levels (TRLs) were introduced and used by different agencies and institutions in their technology development programs.

The third part of the paper develops a variant of the TRL system that is relevant to the smart grid, and in particular to the software that must be employed. The example of GridLAB-D is presented. The relationship between TRL and testing is discussed.

Finally, it is argued that the TRL sequence leaves a trail of evidence that can be used to overcome a well-established risk aversion. The paper should thus be of value to implementers of the smart grid, as a way to become more confident of the software status.

2. The Smart Grid

In economics, a “natural” monopoly can be defined as an “industry in which multiform production is more costly than production by a monopoly” (Baumol, 1977). The situation can arise because the fixed cost of the capital goods is so high that an economy of scale can be arranged so that it is not profitable for a smaller second firm to enter and compete. To prevent utilities from exploiting their monopolies with high prices, they have typically been regulated by government. Utilities such as water, electricity, and natural gas used to be protected monopolies, guaranteed a certain rate of return on their investment. As a result they developed entrenched risk-avoidance behavior, and resistance to innovation. However, the notion that the power companies were a natural monopoly began to be questioned in the 1970s (The Economist, 1998). The idea that there were limitless economies of scale was thought no longer valid, and the point was made law by the Public Utilities Regulatory Policy Act of 1978, that *required* power companies to buy energy from qualifying competing facilities.

Nowadays, electricity has undergone a period of deregulation, and the generators of electric power can now compete. But the infrastructure, i.e. the wires that carry the electricity, usually remains a natural monopoly, and the various companies send their electricity through the same grid. To accommodate a number of new kinds of generation, a *smart grid* is being created. The smart grid essentially takes an electricity grid and intelligently integrates communications and computer technology, so that (for example) suppliers can deliver electricity to consumers in a wider range of conditions, while also accommodating wind and solar power sources.

Thus, utilities in the power sector deal with a shift towards smart grid and energy efficiency. Increasingly, smart grid projects include software-based efforts. Compared to hardware endeavors, a major advantage of this type of projects is a shorter time to market.

Just as with hardware, there is a real need for a metric to measure the technology maturity and integration level of the software in the smart grid power system projects. A scale called Technology Readiness Levels (TRLs), adapted from other fields, is proposed. Because of the documentation required, the metric can be used to increase the comfort-level of utilities as they adopt the new directions, and take advantage of commercialization.

2. Technology Readiness Level and Software Quality

In any project there are multiple interdependent constraints that influence the project success. Three crucial ones are scope, time, and budget. However, many projects somehow get started without sufficient previous planning, and the work-load seems to increase unexpectedly as the projects progress.

According to Donna Shirley (2010), manager of the Mars program at the Jet Propulsion Laboratory, the business of TRLs got started at National Aeronautics and Space Administration (NASA) because of a "guy named Werner Gruel, who was a NASA cost analyst." Gruel "had this great curve that showed that if you spent less than 5 percent of the project cost before you made a commitment to the cost, that it would overrun. If you spent 10 percent of the cost before, it wouldn't overrun." Figure 1 is an updated version of Gruel's graph, showing the overruns on some actual space missions.

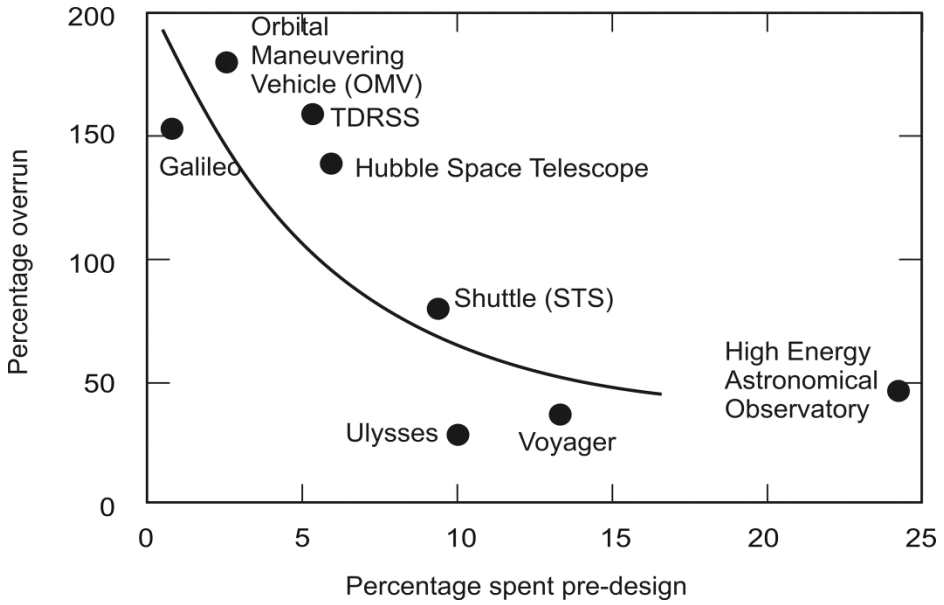

Figure 1: Cost overrun as a function of pre-design spending

What was going on, according to Ms. Shirley, was that at the start of a project, there was a need for a cost estimate, so the engineers did a quick-and-dirty design that they called a "point design." A cost estimate based on the point design was then put into the rather long NASA budget cycle.

However, the chances were good that the point design did not take everything into account. After the funding has been obtained, and work had started, it was gradually realized that there was going to be a cost overrun. At this point, the choices were between an overrun and a descope. Nobody wants to descope, so the result was usually that there was a cost overrun, and the engineers got the blame. What NASA needed was a way to improve their understanding of what was involved. This is where the Technology Readiness Levels came into their own.

TRLs had been under development at NASA since the 1970s, and the curve generated by Werner Gruel provided the ammunition to show they could be useful. The original TRL concept came from Stan Sadin at NASA, in 1974, with seven levels identified (Sadin 1974). The method was developed sporadically after that. The current nine-level version of TRLs was written down in a white paper by John Mankins at NASA in 1995, and seems to have become more or less fixed. Since then, the European Space Agency has adopted something very similar. So has the US military. (The US Air Force has even created a spreadsheet to help determine the TRL of technology. So has the Department of Homeland Security.)

The TRL definitions have some variation in interpretation to suit different organization's needs, but their overall scales match NASA's traditional scale quite closely, focusing on how ready a

technology/integration/system is for its application in the intended environment. In Table 1, Weiping and colleagues (Tan et al. 2009) compare the TRLs used by the leading government agencies in their technology development programs. The agencies tailored the TRL process specifically to their organizations in order to produce operational systems on schedule and within budget. The NASA TRLs are the ones listed by Mankins (Mankins 1995), and the Department of Defense (DoD) ones are from the DoD Deskbook (DoD 2005). The Department of Energy (DOE) TRLs here apply only to the nuclear fuel side of DOE (Carmack and Pasamehmetoglu 2008). NATO TRLs are given in a 2006 document (NATO, 2006).

Table 1: TRL definitions used by the leading government agencies

TRL	National Aeronautics and Space Administration (NASA)	Department of Defense (DOD)	Department of Energy (DoE)	North Atlantic Treaty Organization (NATO)
0	N/A	N/A	N/A	Basic research with future military capability in mind
1	Basic principles observed and reported	Basic principles observed and reported	Initial concept verified against first principles and evaluation criteria defined	Basic principles observed and reported in context of a military capability shortfall
2	Technology concept and/or application formulated	Technology concept and/or application formulated	Technical options evaluated and parametric ranges are defined for design	Technology concept and/or application formulated
3	Analytical and experimental critical function and/or characteristic proof-of-concept	Analytical and experimental critical function and/or characteristic proof of concept	Success criteria and technical specifications are defined as a range	Analytical and experimental critical function and/or characteristic proof of concept
4	Component and/or breadboard validation in laboratory environment	Component and/or breadboard validation in laboratory environment	Fuel design parameters and features defined	Component and/or breadboard validation in laboratory/field (eg ocean) environment
5	Component and/or breadboard validation in relevant environment	Component and/or breadboard validation in relevant environment	Process parameters defined	Component and/or breadboard validation in a relevant (operating) environment
6	System/subsystem model or prototype demonstration in a relevant environment (ground or space)	System/subsystem model or prototype demonstration in a relevant environment	Fuel safety basis established	System/subsystem model or prototype demonstration in a realistic (operating) environment or context
7	System prototype demonstration in a space environment	System prototype demonstration in an operational environment	All quantification steps completed and fuel is licensed	System prototype demonstration in an operational environment or context (eg exercise)
8	Actual system completed and “flight qualified” through test and demonstration (ground or space)	Actual system completed and “flight qualified” through test and demonstration	Reactor full-core conversion to new licensed fuel completed	Actual system completed and qualified through test and demonstration
9	Actual system “flight proven” through successful mission operations	Actual system “flight proven” through successful mission operations	Routine operations with licensed fuel established	Actual system operationally proven through successful mission operations

When first invented or conceptualized, most new technologies are not suitable for immediate application. They must go through a number of stages including experimentation, refinement, and extensive testing, in order to be proven ready for system integration and/or commercialization. As the TRL number gets higher, the range of applications gets narrower. At very low TRL numbers, there could be many applications for a particular technology. TRL 2 is not reached until there is one application in mind. As the TRL number gets higher, the cost of moving from one level to the next gets higher. This cost increase is brought about by the need to involve more specialized workers, and the need for keeping track of things with more documentation.

Figure 2, adapted from Nolte's "whale-shaped" chart (Nolte 2008), shows how the usefulness of a particular technology evolves in time, and how the technology development phases relate to the TRLs. It is clear that most of the TRLs occur early in the technology life cycle, where:

- TRL 1 to TRL 3 address general conceptual science matters,
    - a step up from TRL1 to TRL2 shifts ideas from pure to applied research,
- TRL 4 to TRL 5 cover the transition from scientific research to engineering and system development,
    - TRL 4 is the first step in determining whether the individual components will work together as a system (DOE 2009),
- TRL 6 to TRL 9 focus on engineering matters,
    - TRL 6 begins true engineering development of the technology as an operational system,
    - TRL 9 represents the final stage of the technology, when the technology is fully operational and its maturity is reached.

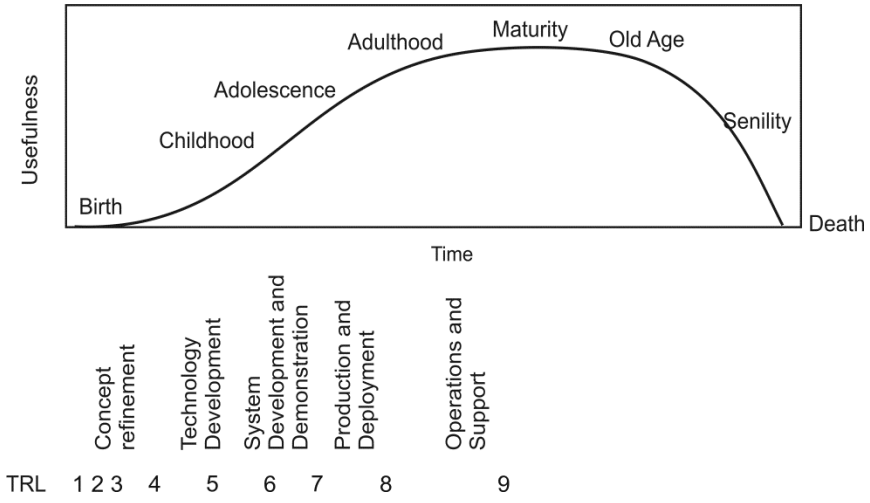

Figure 2: Technology Life Cycle and Technology Readiness Levels

3. Software Technology Readiness Level in Smart Grid

The TRL metrics are applicable to any type of technology, including hardware and software products. Measuring the readiness of a software product reflects some combination of quality characteristics estimated at a given moment of time. However, it is impossible to produce systems of any size which do not change as they develop, so the TRL number is a transient thing. Both the hardware and software environments surrounding a software product change and therefore the software products are continuously changing and aging (Eick et al. 2001).

Once software is put into use, new requirements emerge and existing requirements change, for example as the business running the software changes. Parts of the software may have to be modified to support architectural changes or to correct errors that are found in operation, improve its performance, or other non-functional characteristics. Consequently, even after delivery, software systems evolve in response to demands for change.

Building the smart grid requires a lot of computer and software subsystems. Utilities begin to address the strategy, management, and regulatory issues involved in transitioning to a smart grid. Consequently, the utilities might benefit from using TRLs in their technology acquisition processes.

For hardware, implementing TRLs for the smart grid is not a difficult thing to visualize. The manufacturer of a widget intended for the smart grid can assemble convincing evidence during the design and development process that the widget is at a level of development that it would not endanger the power system if it were put into the field. It has to be admitted that, in the past, due to the lack of this kind of documentation, some technologies have been applied to the electric power system before they were really ready. For example, back in the 1960s, some early applications of solid state technology to power system relaying were less than adequate. Had the devices been subject to the rigor of an appropriate qualification program, that experience could have been avoided. However, because the utilities did have this bad experience, future technology diffusion has to make the effort to be convincing. Hasty adoption of unready technology can sometimes lead to delays in the adoption of ready technology.

For software, the case of an application developed at Pacific Northwest National Laboratory (PNNL) is considered. To assure the correct functionality of some products used in the distribution part of the smart grid, a computer model simulator, called GridLAB-D, was developed at the U.S. Department of Energy's Pacific Northwest National Laboratory. The sophisticated computer program provides a detailed, simultaneous simulation of the electric grid, including power flow, end-use loads, and market functions. In other words, it models both technical and non-technical interactions within a power grid. The model allows users to evaluate new technologies and operational strategies, to craft and refine the characteristics of these technologies and strategies, and to predict the results of deploying them. GridLAB-D was designed as an open source system having contributors all over the world. Modeling in GridLAB-D uses specific classes of objects, and modules, classified by functionality.

The GridLAB-D project team began using TRLs in January 2011 to get a better understanding of technology status, manage risks and make decisions in funding, development and deployment. Moreover, TRLs assist with assessing the readiness of modules and classes for analysis projects and studies. The software TRLs, tailored for the GridLAB-D project, are detailed in Table 2. To assist with an effective and consistent use of TRLs in Table 2, a methodology of software readiness assessment is designed. Unfortunately, the procedural steps and the evaluation of the obtained results are not yet in final form, since the design process is constantly optimized and improved.

Table 2: Software Technology Readiness Level

TRL	Definition	Description
1.	**Basic principles described** (mathematical formulation)	Lowest level of software readiness. Basic concept begins to be translated into applied research and development by providing a detailed mathematical formulation. Examples might include a concept that can be implemented in software, or analytic studies of an algorithm's basic properties.
2.	**Application concept formulated** (algorithm)	Development has begun. Basic individual algorithms or functions are prototyped and documented. Results are speculative and there is no proof or detailed analysis to support assumptions or expectations. Examples are still limited to paper studies.
3.	**Analytical proof of concept** (prototype)	Active research, development and documentation are initiated. Depending on the size and complexity of the implementation, basic components of the integrated critical system have been designed, built and partially tested. Analytic studies to produce code that validates analytical predictions of separate software elements of the technology are done. Examples include implementation of software components that are not yet integrated or thoroughly tested, but satisfy an operational need. Algorithms are run and tested on a surrogate processor in a lab environment.

4.	**Standalone component validated** (earliest version)	Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and reliability compared to the eventual system. System software architecture development initiated to include interoperability, reliability, maintainability, extensibility, scalability, and cyber-security issues. Software integrated with simulated current/legacy elements as appropriate. Verification and validation process is partially completed, or completed for only a subset of the functionality in a representative simulated laboratory environment. Documentation includes design documents and a start of a user manual.
5.	**Integrated component validated** (ALPHA version)	Reliability of software ensemble increases significantly. All software components are integrated with reasonably realistic supporting elements so that the software can be tested and completely validated in a simulated environment. Examples include "high fidelity" laboratory integration of software components. System software architecture is established. Algorithms run on a processor(s) with characteristics expected in the operational environment. Software releases are "Alpha" versions and configuration version control is initiated. Full documentation according to the applicable software standards, test plans and application examples, including all use cases, cyber-security and error handling should be provided.
6.	**System - subsystem demonstrated** (BETA version)	Represents a step up from lab scale to engineering scale. Representative model (BETA version), which is well beyond that of TRL 5, is tested in a relevant environment. Examples include testing a prototype in a live/virtual experiment or in a simulated operational environment. Algorithms running on the simulated operational environment are integrated with actual external entities. Configuration control and quality assurance processes are fully deployed. Verification and Validation process is completed for the intended scope (including robustness) and the system is validated in an end-to-end fully representative operational environment (including real target).
7.	**Prototype demonstrated** (product RELEASE)	Requires the demonstration of an actual system prototype in an operational environment. Algorithms running on processor of the operational environment are integrated with actual external entities. Software support structure is in place. Software releases are in distinct versions. Functionality and performance are not significantly degraded by frequency and severity of software deficiency reports. Verification and validation is completed, validity of solution is confirmed within intended application. Requirements specification are validated by the users. Engineering support and maintenance organization, including helpdesk, are in place.
8.	**System "analysis qualified"** (general product)	Software has been demonstrated to work in its final form and under expected conditions. In most cases, this TRL represents the end of system development. Examples include test and evaluation of the software in its intended system to determine if it meets design specifications. Software releases are production versions and configuration controlled, in a secure environment. Software deficiencies are rapidly resolved through support infrastructure. Full documentation including specifications, design definition and justification, verification and validation (qualification file), users and installation manuals, training and education materials, software problem reports and non-compliances should be provided.
9.	**System proven** (live product)	Represents actual application of the software in its final form and under designed conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the last "bug fixing" aspects of the system development. Examples include using the system under operational design conditions. Software releases are production versions and configuration controlled. Frequency and severity of software deficiencies are at a minimum. Sustaining engineering, including maintenance and upgrades, updates to documentation and qualification files are in place.

The assessment of software readiness level required individual estimations and evaluations (a subject matter expert assesses the TRL of the technology). This is followed by group estimations in meeting or conference format for discussing the technology maturity, and a combination of the above when a

consensus of a single estimate is sought. For each GridLAB-D class, the frequency distribution of the individual estimations and evaluation is constructed and incorporated in the mean value calculation. The method of determining TRL of a class is difficult to be extended to modules because classes can often interact in ways that are not measured by the class TRL. The TRL of each GridLAB-D module is calculated as the lowest TRL found in the classes used by the model. Since the interaction between classes is not quantized in the TRL grading, the actual TRL of any model may be lower than the TRL computed by GridLAB-D. For illustration purposes, Table 3 presents the TRLs assessment for two GridLAB_D models - climate and residential. By choosing what classes are included in a specific model, the user has the possibility to increase or decrease the TRL of that model, i.e. a *residential* home having *house_e* and *water heater* as components has a modul TRL of 7, while *house_e* .by itself has a module TRL of 9.

Table 3: GridLAB-D TRL assessment

Module	**Classes**		**Module TRL**
	Name	**TRL**	
climate	*weather*	*9*	*9*
residential	*clothes washer*	*2*	*2*
	dishwasher	*2*	
	dryer	*2*	
	evcharger	*4*	
	freezer	*5*	
	house_a	*6*	
	house_e	*9*	
	lights	*8*	
	microwave	*5*	
	occupant load	*8*	
	plug load	*8*	
	range	*5*	
	refrigerator	*5*	
	water heater	*7*	
	zipload	*9*	

All of the above proved that TRLs can result in more reporting, paperwork and review time. It takes time for participants in GridLAB-D work to adjust to using TRLs and for the TRL process to have an effect project-wide. Understanding and communicating the detail of the TRL assessment is vital to avoid unnecessary concerns. For example, some elements may have a low TRL but a clear development path and therefore their maturation is a low risk.

Systems engineering processes are not addressed in the lower TRLs, which can result in difficulties transitioning technologies to higher TRLs. There are situations when moving a TRL up on the scale is a matter of providing resources to rapidly increase the TRL or seek an alternative solution (technology) with a higher TRL.

It may be noted that the TRL scale of Table 2 does not exactly parallel the TRL values in Table 1. For example, in the (hardware-based) Table 1, TRL 4 and 5 are no more than breadboard systems, and it is sometimes said that one can reach TRL 6 in a good Hobby Shop. (Mind you, TRL 5 at NASA requires that a failure modes and effects analysis (FMEA) is performed, not something usually associated with a Hobby Shop.) In contrast, TRL 5 in Table 2 talks about “releasing” software, a process that surely involves leaving the laboratory or workshop environment. This difference in process results from the difference in testing needs between hardware and software.

Based on the TRL, stating how successful each subsystem of the hierarchy will be requires that the TRLs be linked to test plans and trials. This linkage provides a clear statement on the TRL achievement at each stage of the project. The TRL definitions provide a convenient means to further understand the

relationship between the scale of testing, fidelity of testing system, and testing environment and the TRL (DOE 2009). As it can be seen in Table 4, the scale requires that for a TRL 6 testing should be completed at an engineering or pilot scale, with a testing system fidelity that is similar to the actual application. TRL 6 represents the point when the software is delivered to customers for beta testing. Therefore, the verification and validation process should be completed for the intended scope and the system should be validated in a simulated environment with some external features. The TRL scale used in Table 2 requires that testing of a prototypical design in a relevant environment be completed prior to incorporation of the technology into the final design of the facility.

The Technology Readiness Level is, of course, not just monitoring for the sake of monitoring. It is a useful part of the management process. TRL levels were originally intended as a way to assess the amount of effort needed to get something qualified for space use. Implicit in that statement is the fact that if a widget was not at a good solid TRL 6 or even a TRL 7, it was *not* going into space. Whether for manned-space or not, these levels were important. In adapting the idea to software, it is not unusual to release software to the users in a somewhat unprepared state. What this means is that at a TRL that would just about get a widget into space, the software is released to Beta test. Therefore, the important TRL number is really TRL 6. What is needed up to TRL 6 controls whether the software leaves the workshop. This is the message to take away from the TRL numbers in Table 2.

Table 4: Relationship of Testing Recommendations to the TRLs

TRL Level	Scale of testing [1]	System Fidelity [2]	Environment [3]
1		Paper	
2		Paper	
3	Lab	Pieces	Simulated
4	Lab	Pieces	Simulated
5	Lab/Bench	Similar	Simulated
6	Engineering/Pilot Scale	Similar	Relevant
7	Full	Similar	Operational (limited range)
8	Full	Identical	Operational (full range)
9	Full	Identical	Operational (full range)

Numbered notes

1. Scale of testing
- Full Scale matches final application.
- 1/10 Full Scale < Engineering/Pilot Scale < Full Scale (Typical)
- Lab Scale < 1/10 Full Scale (Typical)

2. System Fidelity
- Identical – matches final application in all respects
- Similar – matches final application in almost all respects
- Pieces – matches a piece or pieces of the final application
- Paper – exists on paper

3. Environment
- Operational (full range) – full range operational capacity
- Operational (limited range) – limited range operational capacity
- Relevant – simulated environment plus a limited range of external features
- Simulated – restrictive range of simulation

4. Conclusions

Properly applied in system engineering management, Technology Readiness Levels are of value in understanding technology status, managing risks and making decisions in funding. For the development and deployment of software products they assist the system engineer and the manager. The TRLs reduce the amount of subjectivity about project status, and therefore, planning and execution can be managed better.

TRLs can serve a number of useful purposes in the electric generation and delivery world. Overall, the transitioning of technology from laboratory to application becomes a carefully managed process. For the utility world and the hardware headed for the smart grid, the TRL sequence leaves a trail of evidence that can be used to overcome a well-established risk aversion.

The same observation is necessarily true for software. However, for software there are a few additional difficulties in applying the standard TRL method. For example, inevitable software “decay”, architecture transformations and the need for software maintenance after release, increase the volume of documentation involved. Nevertheless, the documentation serves a purpose, and it can be argued that it justifies its existence.

The proposed software TRLs described in this paper have been applied to an actual software system development, i.e. GridLAB-D. While the application of TRLs to the software is presently in its relatively early stages, it strengthens the smart grid software capabilities even further by pinpointing and even preventing system malfunction before release to the customers.

References

DOD 2005 May. Technology Readiness Assessment (TRA) Deskbook,

DOE G 413.3-4 2009, October. US DOE Technology Readiness Assessment Guide,

The Economist 1998, March. Power to the people: Deregulation and new technology are working hand in hand to transform the global electricity-supply industry,

Eick, S., Graves, T., Karr, A., Marron, J., and Mockus,A. 2001. Does Code Decay? Assessing the Evidence from Change Management Data. IEEE Transactions on Software Engineering, Vol. 27, No. 1,

Mankins, John C. 1995, April. Technology Readiness Levels: A White Paper. NASA, Office of Space Access and Technology, Advanced Concepts Office,

NATO 2006. North Atlantic Treaty Organization (NATO) Technology Readiness Levels. http://www.nurc.nato.int/research/trl.htm,

Nolte, William L. 2008. Did I Ever Tell You about the Whale? or Measuring Technology Maturity. Information Age Publishing,
,

Sadin, Stan 1974. Origin of TRL. http://en.wikipedia.org/wiki/Technology_readiness_level
Shirley, Donna 2010. www.jsc.nasa.gov/history/oral_histories/NASA_HQ/.../DLS_7-17-01.pdf,

Tan, Weiping, Ramirez-Marquez, Jose, and Sauser, Brian 2009. A Probabilistic Approach to System Maturity Assessment. Wiley Online Library, DOI 10.1002/sys.20179.