Costs of Achieving Software Technology Readiness

Introduction

Many of the complex systems required by the Department of Defense (DoD) and NASA require technological innovations to achieve sophisticated missions and state of the art accomplishments. Technology Readiness Assessments (TRAs) for complex systems and subsystems are important for both the government and its contractors to ensure proposed technologies meet program requirements. Without good technology assessments, programs may be funded that have little or no chance of success because success assumes use of technologies not yet confirmed to be realistic. The defense acquisition community recognizes this and has developed detailed guidelines for Technology Readiness Assessments (TRAs). [1]

Wikipedia defines Technology Readiness Level (TRL) as a measure used to assess the maturity of evolving technologies prior to incorporating that technology into a system or subsystem [2]. The US Department of Defense (DoD), NASA and many large corporations use TRLs to determine whether the design of systems is technically feasible prior to the start of any significant design or development activities. Both the DoD and NASA have developed TRL scales from 1 to 9 to facilitate technology maturity assessment. A level 1 represents technology that still resides on cocktail napkins and in laboratory experiments while a 9 represents technology that has been deployed successfully in systems currently fielded.

As new technologies are required and proposed, the cost community becomes an important component to program success. A good cost estimate for a program relying on currently immature or non-existent technology should include the costs associated with bringing that technology to a maturity level of 6 or greater.

One problem frequently cited with the definition and application of TRLs (as defined by both the DoD and NASA) is that they are very focused on hardware. And much success has been realized with their application in the hardware world. However, software is different than hardware. It's not as obvious what software technology means and how it is best assessed. Software brings almost infinite possibilities for advancements of the state of the art, but these possibilities require the right mix of hardware, tools, people and processes. Assessing the state of software technology becomes a 'softer' exercise. Consequently, the issues associated with estimating the costs of transcending Software TRLs become less obvious as well.

This paper starts with a brief discussion of technology readiness levels (TRLs) and technology readiness assessments (TRAs). It then defines the components of software technology as they relate to the assessment of technology maturity. This paper

presents the factors that may complicate software TRAs and offers guidance on gaining a better handle on their impacts. Finally, a methodology is presented to give cost estimators and analysts the tools to perform credible cost estimates that incorporate the effects associated with advancing software technology to appropriate levels of maturity.

TRLs and TRAs

For most of us the concept of technology readiness is hard to grasp. This is because in general, our experiences with technology are with fully matured technology. In 1961, President Kennedy challenged US scientists, mathematicians and engineers when he announced that within the decade of the 1960s the US would 'land a man on the Moon, and return him safely to Earth'. At the time, there were no solutions to solve problems such as reaching earth's orbit and traveling to the moon, let alone for how man would be able to survive in space. Some very smart people and creative thinking was needed to invent solutions that didn't exist in order to make Kennedy's dream a reality.

One of the many things that these smart people discovered was that programs envisioned when the technology is very new or non-existent are much harder to plan for than programs using technology that has been used successfully on other programs. As the possibilities for the space program greatly outsized the budget for space programs this fact became increasingly problematic. In 1974 Stan Sadin of NASA developed Technology Readiness Levels as a methodology to assess the technology readiness of a proposed spacecraft design. Eventually the methodology was institutionalized by NASA and later similar methodologies were developed by the US Department of Defense (DoD) and other organizations responsible for the acquisition of complex aerospace and defense programs. TRLs are used by NASA and the DoD to support go/no-go decisions at various acquisition milestones. Figure 1 summarizes NASA's TRL view. Table 1 summarizes a generalization of TRLs used by DoD and NASA.

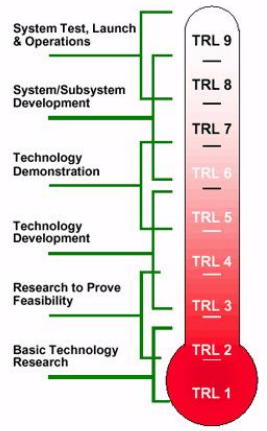

Figure 1 : NASA's TRL Meter

	TRL	Description
1	Basic principles observed and reported	Scientific research begins to be translated into applied research and development - "back of the napkin sketch"
2	Technology concept and or application formulated	Analytical studies are conducted - "confirm that idea is good and useful"
3	Analytical and experimental critical function and/or characteristic proof of concept	Active resarch and development begins, physcial validation of analytical predictions - "prove idea is possible"
4	Component and/or breadboard validation in laboratory environment	Basic technology components are integrated to establish that they work together - " prove idea presents realistic solution"
5	Component and/or breadboard in a relevant environment	Basic technology components are integrated with reasonably realistic supporting elements for test in a simulated environment - "alpha version of technology"
6	System/subsystem model or protoype demonstrated in a relevant environment (ground or space)	Model or prototype of solution is tested in a relevant environment - "beta version of technology"
7	System prototype demonstration in operational environment	Demonstrattion of an actual system prototype in an operational environment - "release candidate ready for operational test"
8	Actual system complete and quaified through test and demonstration	Technology has been proven to work in its final form and under expected conditions - "Technology has gone gold"
9	Actual system proven through successful mission operations	Actual application of the technology in its final form and under mission conditions such as those encountered in operational test and evaluation - "technology has been used successfully in target environment

Table 1 : TRL Descriptions (generalized)

The intent of the TRA is to document that prior to system design and development, there is a reasonable expectation that the acquisition is technically feasible. [1] The DoD has outlined a detailed process for executing a TRA which is documented in [3]. TRA's are required for any Acquisition Category 1 (ACAT1) programs at Milestones B and C. Figure 2 shows how TRLs and TRAs integrate into the general acquisition process. [3]. The TRA process is summarized below.

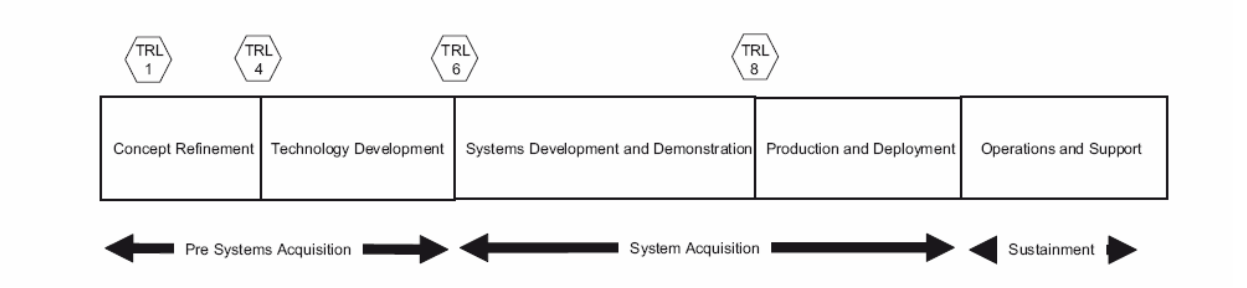

Figure 2: Acquisition process perspective for TRLs

The basic approach for a TRA requires identification of the critical technology elements (CTEs), those items for which the technology is new and novel or where technology will be expected to operate in environments which are different from any previous operational environment for that technology. It is generally recommended that the Work Breakdown Structure (WBS) be used as a guide for CTE identification although there are problems with this approach when dealing with software. Each CTE is evaluated against the definitions of the technology readiness level and assigned a level. This examination could include study of program concepts, requirements for technology, and actual demonstrated technology capabilities. Maturation plans are then developed for any CTE that is not mature.

Software TRL

There are two aspects to software technology – the technologies and tools that support the development of software for operational systems and the algorithms that are implemented in an operational system using software. Robert Gold [5] discusses the five distinct ways that software can be evaluated for technology readiness

1. **Unprecedented Functionality**

   Software implements algorithms or methods developed by domain experts. To assess readiness it is important to understand whether the algorithms or methods are new or novel.

2. **Off the shelf components**

   While it is common to make an assumption that off the shelf solutions are mature since they are already being used commercially, it is not always true. While the capability may not be new or novel, the capability may not have been proven in the intended operational environment

3. **Enabling Run Time**

   There are technologies that make up a backbone of a system that may be transparent to end user but are required to make the software operate correctly. Such capabilities, when vital to the successful operation of the software system should be evaluated as potential critical technology elements.

4. **Aggregation of components**

It is important to understand how various software components need to be aggregated to work together to create system capabilities. Components that may not be critical on their own may gain criticality if their interactions prevent critical components to succeed. It may be necessary to evaluate criticality at a system or subsystem be evaluated. As noted in [6], "two aspects of software are, by definition new… The systems as a whole and the interfaces"

5. **Enabling Development**

   In addition to the maturity of the algorithms and methods, and the supporting hardware and software, it may also be necessary to evaluate the maturity of technologies critical to development and evolution of the system which are not deployed with the system. Basically this category includes programming, management or process tools that support the development and maintenance of a system.

Software TRL Challenges

It's clear from the terminology of the TRLs that the original focus for technology readiness assessment focused primarily on hardware. Software is, of course, different than hardware. The Army has developed software specific TRLs (Table 2) which help with some issues but challenges remain that are very specific to software and how it is used in programs. Software brings almost infinite possibilities for advancements of the state of the art, but these possibilities require the right mix of hardware, tools, people and processes. Assessing the state of software technology presents unique challenges. These challenges include dealing with issues surrounding off the shelf capability, identifying and dealing with critical technology associated with supporting hardware and software and issues associated with integration and interoperability.

Software applications fall into one of two categories: embedded software and Information Technology (IT) applications. Embedded software uses IT components to deliver specific unique capability required for successful operation. This is the type of software that is developed for weapons systems, aircraft, etc. Embedded software is generally composed of new capability integrated with capability that has been reused from similar applications. The software system and the interfaces should both be considered as potential CTEs. Generally for these systems IT is not considered a

critical element in the delivery of the software unless it is new or subject to unique requirements in the context of the software.

	TRL	Description
1	Basic principles observed and reported	Basic reseaqrch begins to be translated into applied research and development. Examples include concept that can be implemented in software or an anlytic study of algorithms basic properties
2	Technology concept and or application formulated	Analytical studies are conducted
3	Analytical and experimental critical function and/or characteristic proof of concept	Active resarch and development begins, this includes analytical studies to produce code that validates analytical predictions of software elements. Examples include software component not yet integrated or algorithms run on a surrogate processor in a lab
4	Component and/or breadboard validation in laboratory environment	Basic software components are integrated to establish they will work together. Components are primitive with respect to efficiency and reliability . System software architecture development initiated
5	Component and/or breadboard in a relevant environment	Reliability of software ensemble increases significantly. Basic software components are integrated with reasonably realistic support elements for tests in simulated environment. System software architecture established. Software releases of alpha version quality
6	System/subsystem model or protoype demonstrated in a relevant environment (ground or space)	Representative model or prototype system, is tested in relevant environment. Examples include testing a prototype in a live/virutal experiment or in simulated operational environmen
7	System prototype demonstration in operational environment	Demonstrattion of an actual system prototype in an operational environment. Algorithms run on processor of the operational enviroment integrated with actual external entities
8	Actual system complete and quaified through test and demonstration	Software has been proven to work in its final form and under expected conditions. Examples include test and evaluation of the software in its intended system to determine if it meets design specifications
9	Actual system proven through successful mission operations	Actual application of the software in its final form and under mission conditions such as those encountered in operational test and evaluation

Figure 3 : Summary of Army Software TRL Definitions

IT applications are composed of capabilities that are available off the shelf. These applications rely mostly on off the shelf technology to meet system capability requirements. The benefit of off the shelf components include potential cost and time savings as well as the promise of software that has been tested through use in the field. These benefits can be realized but there are other areas for consideration. Even if the

capability is not new and novel and has been field tested, if it has not been field tested in the operational environment of the target system it requires consideration as a CTE. Additionally, there are issues associated with the fact that software never stops changing. According to [7], Commercial Off The Shelf (COTS) products generally undergo a new release every eight to nine months with active vendor support for only the three latest releases. Is it safe to assume that upgrades do not incorporate new or novel algorithms or is re-evaluation necessary? Further, COTS vendors sometimes retire their products, introducing a new or improved alternative or sometimes presenting no alternative at all. A COTS product with a known 'end of life' could present as a CTE if it's capability is critical or if it supports critical capability.

Unlike hardware, software cannot stand alone. Without hardware and supporting software, a software application is useless. Evaluation of criticality needs to consider all of the components required for successful delivery. Software that is new and novel should be a CTE. The Information Technology (IT) infrastructure that supports the software should be evaluated as potential CTE as well. Additionally, there may be criticality associated with conversions of legacy tools or environments if backward compatibility is crucial.

Software TRL Costs

In order to estimate the costs to advance from a "cocktail napkin" to fielded capability it is necessary to understand what the CTEs and their TRLs are. What follows is a methodology for tailoring a parametric estimating methodology to estimate the journey from TRL 1 to TRL 9. This methodology addresses consideration of cost drivers relevant to technology maturity issues.

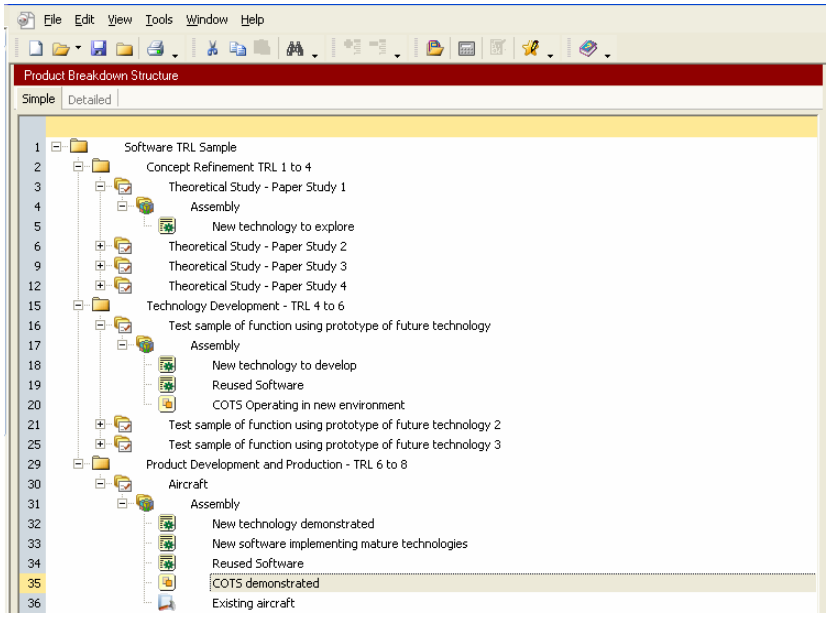

Figure 4 : Sample Breakdown for Software TRL estimates

Figure 3 provides a sample product breakdown structure for estimating software technology maturity. The analysis is partitioned into three parts.

- Costs for theoretical study to prove the concept (TRL 1 – 4)
- Costs for development of the technology (TRL 6 – 8)
- Costs for development and production of the system incorporating new technology with existing technology

To transition from TRL 1 to 4 requires paper studies and laboratory experiments. It is important to plan for theoretical studies since multiple aspects of the new technology need vetting. It's likely that going from TRL 1 to 4 will require more studies than going from TRL 3 to 4 so starting technology level is relevant. At TRL 1 and 2 the activities are focused on algorithm validations and high level design. Only activities such as concept, requirements and high level design are pertinent for TRL 1 and 2, for TRL 3 and 4 some code and unit test should also be relevant.

Code size should indicate only the 'size' of the technology being considered and should be estimated based on analogous experience with similar technologies. Complexity values for the code size tend to be higher than organizational norms since innovation generally trends to the more complex. Even if the technology is eventually targeted for space or airborne equipment, this does not need to be accounted for in early studies as these are experimental and not intended for practical use. Cost drivers indicating the amount of new design should be set to high for the new technology elements.

The move from TRL 4 through 6 will require focus on additional activities as actual development of new technology occurs. Coding and unit test of the new technology elements will need to occur. Integration and test activities need to be achieved in order to prove the new technologies play well with mature technologies and legacy capabilities.

Code size for new technology element(s) should reflect the expected 'size' of the implementation based on analogous experience and discoveries during early experiments. Consider how much, if any, code exists from prior studies that can be used to implement this new technology and indicate size as new, modified and reused. Size for COTS being validated in new operating environments should reflect the functions of the COTS required meeting the eventual system requirements. A size should also be determined for the functionality which exists or will be developed that is not part of the technology study but whose integration with the new technologies is a critical technology element for this system. Cost drivers indicating the amount of new

design should still be high, though they should be slightly lower than the earlier analysis if some high level design has already been accomplished.

Since software that may become production software is included in the technology development phase, the eventual operating platform may be a significant cost driver at this point. Additionally, cost drivers associated with integration complexities, development personnel and integration team personnel should be scrutinized as integrations are often an area of high criticality.

Once TRL 6 has been achieved the development of the system becomes a typical project. The cost estimate for this project follows traditional estimating guidelines. Because software implementing the new technology elements already exists this should be modeled as reused or modified code instead of new. Cost drivers indicating the amount of new design should be lowered for the new technology elements since this design has already been developed and validated. Where relevant, the cost drivers for integration complexities and integration team personnel and experience should reflect the fact that some integration has already been prototyped and tested.

Figure 5 is an example of the progression of a project through the various levels of technology maturity.

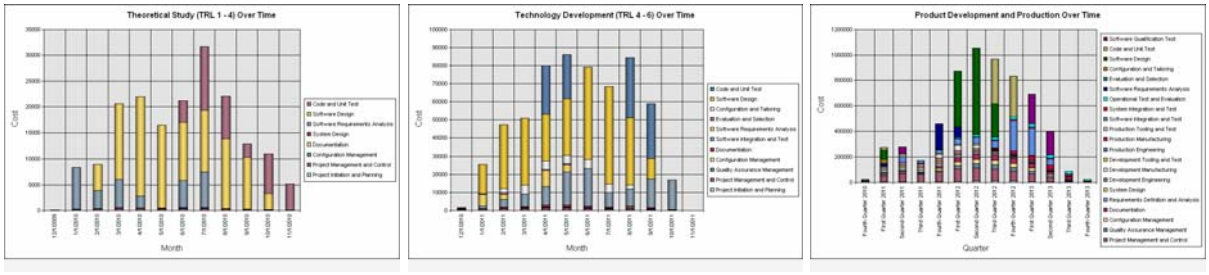

Figure 5 : Plan for the phases of technology maturity

Conclusions

Technology readiness assessments for complex systems and subsystems are important for both the government and its contractors to ensure proposed technologies meet program requirements. Towards this goal, NASA, the US DoD and other Departments of Defense throughout the world have adopted a method of assessing technology readiness with Technology Readiness Levels. While each organization has a slightly different definition for the various levels of TRLs, all represent the same basic steps in the development of new technology. TRLs range from a 1 indicating a very sketchy good idea to 9 indicating technology that has been used successfully in the intended environment.

TRLs have proved useful in helping organizations manage risks specifically associated with technology maturity. From a legacy perspective, TRLs are primarily targeted at the hardware side of acquisition projects. More and more, software requirements are overshadowing hardware requirements in acquisition projects, particularly with respect to requirements being met with new technology elements. Evaluation of TRLs for software is problematic because of the unique aspects of software and software projects. Software requires hardware and other software to operate. This presents challenges in the identification of where critical technology elements reside. Additionally, the use of software Commercial off the Shelf complicates the process of identifying Critical Technology Elements. Interfaces and interoperability of new technology components with legacy and mature technology components presents a potential technology risk.

In cases where immature (or not mature enough) technology is required to meet program needs, effort and cost needs to be expended to mature the technology before the program begins development. Parametric cost estimating techniques can be adapted to estimate the resources required to traverse from TRL 1 to TRL 9 in order to determine the cost and schedule of the entire program. A methodology has been outlined to accomplish this analysis along with guidelines for how to select values for relevant cost drivers. By adapting cost drivers such as software size, amount of new design, integration complexities and code reuse it is possible to model the activities that go on with paper designs, theoretical studies, laboratory experiments and proof of concept exercises focused on the new technology only. Additionally, recommendations have been made as to how to model the final system in light of effort already accomplished to mitigate the technology risks.

References

[1] Gold, R. & Jakubek, D., “Technology Readiness Assessments for IT and IT-Enabled Systems”, presented at Software and Systems Technology Conference, Salt Lake City, Utah, April 2005

[2] http://en.wikipedia.org/wiki/Technology_readiness_level

[3] Technology Readiness Assessment (TRA) Deskbook, Prepared by the Deputy Under Secretary of Defense for Science and Technology (DUSD S&T), September 2003, available at http://www.darpa.mil/STO/solicitations/sn07-44/pdf/TRA_Desktop.pdf

[4] Shermon, D, “Systems Cost Engineering : Program Affordability Management and Cost Control, Gower Publishing, UK, Copyright 2009

[5] Gold, R., “Software Technologies in Technology Readiness Assessments for DoD Acquisition Programs”, Software Tech News, January 2010, Vol. 12, No.4

[6] Winton, D., “Mission Assurance Through Improving Software Technology Readiness Assessments (TRAs) for National Security Space (NSS) Systems, The Aerospace Corporation, TR-2006 (8550)-1, available at (Retrieved March 2010): http://www.ccpe.csulb.edu/spin/media/pdf/Mission%20Assurance%20Through%20Improving%20Software%20Technology%20Readiness%20Assessments%20(TRA)s%20for%20National%20Security%20Space%20(NSS)%20Syste.pdf

[7] Smith, J.D., “An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI)”,Proceedings of the 38th Hawaii International Conference on System Science, 2005