

# Readiness Levels for Spacecraft Information Technologies<sup>1</sup>

Ryan Mackey  
Mail Stop 126-147  
California Institute of Technology  
Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109  
818-354-9659  
[Ryan.M.Mackey@jpl.nasa.gov](mailto:Ryan.M.Mackey@jpl.nasa.gov)

Raphael Some  
Mail Stop 198-219  
California Institute of Technology  
Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109  
818-354-1902  
[Raphael.R.Some@jpl.nasa.gov](mailto:Raphael.R.Some@jpl.nasa.gov)

Abdullah Aljabri  
Mail Stop 198-326  
California Institute of Technology  
Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109  
818-354-5862  
[Abdullah.S.Aljabri@jpl.nasa.gov](mailto:Abdullah.S.Aljabri@jpl.nasa.gov)

*Abstract*— The New Millennium Program (NMP) seeks to advance space exploration through the maturation of promising spaceflight technologies. NMP, like many other organizations, relies upon Technology Readiness Levels (TRLs) as a key indication of technology advancement, and assesses development progress against this generalized metric. A given technology's TRL is based upon our ability to predict how the technology will perform in various applications, and therefore depends on the environment in which the technology has been tested and validated. Testing begins in the laboratory, advancing through ever-improving simulations and testbeds, until finally actual in-space validation is achieved. This process is well understood for space hardware and has been applied for many years.

Presented in this paper is a modified interpretation of the traditional TRLs [1] aimed solely at information technologies. The intent of this new set of definitions is twofold: First, to enable a definitive measurement of progress among developing information technologies for spacecraft; and second, to clarify particular challenges and requirements that must be met as these technologies are validated in increasingly realistic environments. The first goal of this paper reflects NMP's need to verify clear and defensible progress of technology development on the path to spaceflight. The second goal serves to answer the question of what technologies require validation in space, and what salient features of the space environment are important to technology developers. To answer this second question, we will revisit the notion of the "relevant environment," an environment that adequately stresses the technology to provide sufficient confidence in the results.

This paper includes a direct comparison between the traditional TRLs and the modified definitions specific to information technologies. We will also discuss two representative examples to illustrate this process.

---

## TABLE OF CONTENTS

---

1. INTRODUCTION
2. OVERVIEW OF TECHNOLOGY READINESS LEVELS
3. INFORMATION TECHNOLOGIES VS. SOFTWARE
4. SPECIAL ISSUES OF INFORMATION TECHNOLOGIES
5. TECHNOLOGY MATURATION VS. IMPLEMENTATION
6. TECHNOLOGY READINESS LEVEL DEFINITIONS
7. EXAMPLES
8. CONCLUSION

## 1. INTRODUCTION

Technology Readiness Levels are a typical yardstick of technology maturity – a separate concept from the maturity of any specific application – used across NASA. This is an abstract scale, arbitrarily chosen (for NMP purposes) to range between 1 and 9, that reflects the development high-water mark of a given technology. The goal of the New Millennium Program is to support development of new technologies up to and including its first proof in spaceflight. Roughly speaking, NMP is most concerned with technologies in the middle of the scale, between about TRL 3 and 7. This refers to the full range of technology development after the formulation of the method, from first laboratory experiments and prototypes up to flight validation.

Information technologies, including not just software and algorithms, but also tools, methods, and approaches of information representation used to develop software, are ever more important to spacecraft and space based science missions. However, information technologies are generally difficult to assess in this fashion. The principal difficulty lies in understanding the environment that is relevant to the information technology, an environment that differs greatly

---

<sup>1</sup> 0-7803-7651-X/03/\$17.00 © 2003 IEEE

from that relevant to space hardware. Testing requirements of temperature, vibration, and radiation tolerance among others give way to operating system, flight processor resources, and other limitations or stresses of the "environment."

We may pose the question as follows: What constitutes an advance in spacecraft and spacecraft-associated software, which we will term Space Systems Information Technology (SSIT)? How should we interpret the NASA TRL definitions when applying them to SSIT?

Information technology (and, by inclusion, SSIT) is fundamentally different from most other technologies in that other technologies have their foundations in the physical sciences -- physics, chemistry, and the like. Information Technology, on the other hand, deals with the representation and manipulation of information, and has its roots in mathematics and related disciplines.

Conversely, IT is similar to other technologies in that it is built on some underlying theory or basic principles. In this regard, there is also an underlying principle for the technology, which provides the basis for accurate predictions of the performance and characteristics of products built using this technology. Like any other technology, as the TRL increases and the technology is matured, our ability to predict the technology performance characteristics improves. We should therefore be able to apply a similar list of criteria to evaluate SSIT maturity.

## 2. OVERVIEW OF TECHNOLOGY READINESS LEVELS

It is important to distinguish between assessment of technology and assessment of any specific engineering effort. Engineering maturity assessment can be thought of as determining whether or not a device works as it should. Technology maturity assessment is more concerned with *why* a device works, and whether we can estimate how the device would work if it was modified or placed in a specific environment. This is more difficult, requiring thorough prediction and validation of the results of a carefully constructed technology validation experiment. A technology with TRL of 7 or higher indicates that the technology is ready for inclusion in a mission, having been flight validated at least once, and therefore it is reasonable to engineer new systems based on the same technology. TRL must be reassessed only when there is a fundamental change in the underlying theory or when applying to a radically different domain.

When we discuss TRL we often speak in terms of a *model* of the technology. The model we need predicts how the technology behaves in specific environments, for instance how a certain semiconductor technology responds to a range of temperatures, voltages, radiation dosages, etc., permitting

a system designer to estimate how any potential design would function in any range of conditions. As the technology improves and TRL increases, this model must become better defined and more thoroughly proven through refinement and testing of the technology. Consequently, the conditions for attaining each TRL follow this process and provide guidelines to technology developers.

An example of NASA TRL guidelines [1] is given below in Figure 1. The most significant element of the TRL guidelines is the *relevant environment*, in which technologies must be tested to reach TRL 5. A relevant environment is a carefully arranged laboratory environment that accurately simulates the key difficulties and stresses of operation in space. It must sufficiently stress the technology to permit an accurate, defensible estimate of how the technology will perform in its intended application.

The New Millennium Program is primarily concerned with TRLs between 3 and 7. Technologies at TRL 3 have been formulated and tested sufficiently to show that they are useful, but have not yet been developed as a complete prototype. As the technology advances in this middle range, technology prototypes are completed, tested in increasingly sophisticated environments, and finally validated in their first test flight.

## 3. INFORMATION TECHNOLOGIES VS. SOFTWARE

IT is a broadly used term describing everything from email systems to the underlying theories of artificial intelligence. For our purposes, i.e. for use by NMP in defining technology development activities, it is important to draw a distinction not only between SSIT and Hardware, but also between SSIT and Software. Hardware and software are merely separate implementation approaches to providing a specific function. Numerous examples of functions exist that can be achieved through either all-hardware or all-software solutions. SSIT is not directly concerned with the function itself, but rather with the methods used to represent and manipulate information, and with the approaches, tools and environment used to develop products that provide the desired functionality. Thus, while SSIT's for NASA missions are inextricably linked with the algorithms they support, we wish to evaluate the maturity of these methods separately from the algorithms themselves.

The maturity level of a SSIT is indicated by the reliability and efficacy of its tools and methods. A SSIT's maturity can therefore be determined by the presence and reliability of methods and tools used to produce products based on this technology, and by the ability of the practitioner to predict the performance characteristics of these products. The tools and methods will vary significantly depending on the technology, its scope, and its complexity, but in all cases there will exist some fundamental underlying principles or

# Hardware Systems

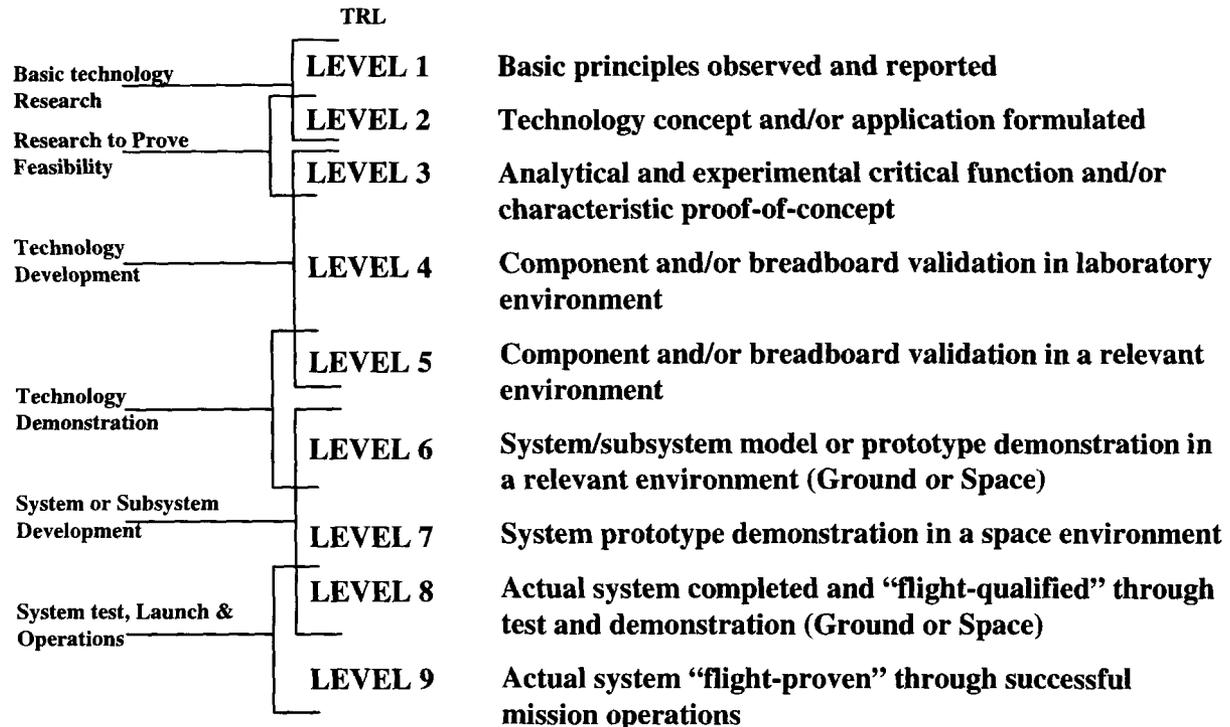


Figure 1: NASA General TRL Guidelines

theory and a set of tools and methods by which products can be built and performance characteristics predicted.

The means and methods to predict performance are the “models” of the SSIT. It is similar in concept to simulations that predict the behavior of mechanical devices. As the technology matures, the accuracy and scope of the model, and concordantly its ability to predict the performance of software products, must improve as well.

## 4. SPECIAL ISSUES OF INFORMATION TECHNOLOGIES

There are two specific issues in SSIT which bear additional discussion: the relatively isolated and insulated nature of the SSIT developer with respect to the rest of the spacecraft and mission development team, and the notion of the SSIT ‘environment.’

A fundamental problem with Spacecraft System Information Technologies and their use in space system software development is the lack of understanding, by many spacecraft developers and managers, of the impact of SSIT decisions on the rest of the spacecraft, and vice-versa. SSIT is rarely designed to firm specifications in a like manner to hardware. Specifications are frequently left in vague terms,

allowing spacecraft designers to exploit software flexibility to meet specific challenges. Historical evidence points to a proclivity on the part of spacecraft developers to implement major changes, using software, with little or no thought to the resulting impact on software development. It is noteworthy that a similar problem is emerging in configurable hardware elements (such as Field Programmable Gate Arrays, or FPGAs) for next generation space systems. Similarly, SSIT developers often do not take into account fundamental differences between the space system environment and their desktop or laboratory environment. All parties must therefore pay special attention to the issue of SSIT impact during the development process. In terms of SSIT TRL, this implies relatively early definition of the required/desired SSIT environment, software and hardware requirements, and availability of tools. The fact that software and “configware” will be used for last minute changes brings with it a need for any SSIT to provide rapid validation and environmental simulation capabilities within their tool suites and methodologies. This need differs from the standard TRL definitions.

The general TRL Guidelines are focused on the relevant environment, meant to indicate a testing environment that adequately captures all of the important environmental features of actual flight. Relevant environment for SSIT, and more specifically for the software products developed using these technologies, is not the physical environment

(radiation environment, microgravity, etc.), but rather the information environment. The SSIT/software product environment comprises information flow patterns, the computing environment (consisting of the instruction set and architecture of the computing hardware), the operating system, the network, and the input/output structures. Environmental stresses include speed of information flow and input/output operations, boundary values, and resource limitations such as available central processing unit (CPU) cycles and memory. The maturation process for SSIT must consider several facets of the environment, as listed below:

*Information Environment* – Includes the hardware (CPU, data bus, etc.), effects of the environment (radiation, lack of maintenance support) on the hardware such as single event upsets (SEUs) and other faults, operating system, supporting libraries, memory and timing constraints, and so on. Because we must specify these constraints before any software products can be meaningfully tested, the information environment must be defined early in development, at TRL 2.

*Spacecraft Environment* – Includes all interactions between SSIT and the spacecraft, such as interfaces, data formats, time stamps, and commands to SSIT. The spacecraft environment must be defined and simulated at TRL 5, as part of the relevant environment.

*Development Environment* – Contains all software tools, debuggers, simulators, etc. needed in order to produce a working SSIT system. An early build of the Development Environment that is sufficiently populated to produce a working prototype must be defined at TRL 3 and completed at TRL 4, coincident with completion of the prototype. Further completion of the Development Environment is not required until after the first flight of the SSIT, at TRL 8 and beyond.

Careful consideration of the SSIT environment also highlights a peculiarity about on-board vs. on-ground technologies. Hardware technologies are unambiguously separable in this regard, but SSIT and its software products can indeed have a dual role. Certain on-ground software, for example, does affect spacecraft performance and behavior, and it is not unreasonable to consider ground and spacecraft SSIT operating as a coherent, indivisible system. Additionally, certain processing functions can be transferred between ground and spacecraft, i.e., can reside in either location.

Another example would be cooperative software working on different members of a spacecraft constellation. It is insufficient to validate only a single part of such a distributed SSIT system. The SSIT environment includes *all* sources of interaction, including ground-based software and interfaces between different spacecraft, when applicable to the particular technology. In systems that are easily

decoupled, such interfaces are easy to simulate. However, cooperative or autonomous SSIT requires special attention to the end-to-end system, which may include ground-based systems.

NMP is not focused on validation of ground technologies, but is concerned with validation of complete space systems.

It is also not unreasonable to propose similar guidelines for maturation of ground-based IT, but a rigorous definition is outside the scope of this document.

## 5. TECHNOLOGY MATURATION VS. IMPLEMENTATION

Many information technologies that will be new to spacecraft will not be new to other domains. It is anticipated that the majority of information technology maturation efforts for spacecraft will be simple cases of technology infusion rather than development.

Because the same requirements on testing and verification apply to infusion as to development, the same guidelines are applicable in either case. In terms of technology maturation and validation, infusion can be considered a case of expanding and validating the bounds of the technology's model and can often be characterized as determining and validating the technologies performance characteristics in a new (relevant) environment and with extended or new environmental stresses. Technologies ready for infusion are typically much further along in development, greatly reducing the time and effort needed to validate tools and benchmark performance. There are also technologies that have been qualified on aircraft or in other stressful environments, but not for spacecraft.

Depending on the technology and the scope of its testing and validation, we may start technology infusion efforts at relatively high TRLs, possibly as high as 5 in the case of technology validated on aircraft. In the majority of cases, we will want to return to low TRLs, but progression through the early stages should be rapid as space-specific documentation and porting/testing in the new environment is completed.

To illustrate this effect, an example of space application of an existing mature information technology is given in Section 7.

## 6. TECHNOLOGY READINESS LEVEL DEFINITIONS

A general list of TRL requirements intended for Information Technologies is given below. This defines the specific tests that must be passed before promotion to the next TRL.

*TRL 1* – Identified/invented and documented a useful information technology with a qualitative estimate of expected benefit. Basic functional relationships of a potential application formulated and shown to be compatible with reasonable processing constraints.

*TRL 2* – Completed a breakdown of information technology into its underlying components and analyzed requirements and interactions with other systems. Defined and documented the relevant IT execution environment. Preliminary design assessment confirmed compatibility with the expected IT environment.

*TRL 3* – Key components of IT prototyped to prove scientific feasibility. Successful preliminary tests of critical functions demonstrated and documented. Experiments with small representative data sets conducted. IT development environment and development tools required to complete prototype defined and documented.

*TRL 4* – Prototype completed on laboratory hardware and tested in a realistic environment simulation. Experiments conducted with full-scale problems or data sets in a laboratory environment and results of tests documented. IT development environment completed as needed for the prototype. A model of IT performance, adequate for prediction of performance in the intended space environment, must be documented as a result of these tests.

*TRL 5* – Prototype refined into a system and tested on simulated or flight-equivalent hardware. Interaction environment, including interfaces to other systems, defined and included in the testing environment. Rigorous stress testing completed in multiple realistic environments and documented. Performance of the IT in the relevant environment must be documented and shown to be consistent with its performance model.

*TRL 6* – System ported from breadboard hardware testbeds to flight hardware and tested with other systems in realistic simulated environment scenarios. IT tested in complete relevant execution environment. Engineering feasibility fully demonstrated.

*TRL 7* – Information technology validated in space. Adequate documentation prepared for transfer from developers to full operations engineering process team.

*TRL 8* – Development environment completed and validated. Approved by an independent verification and validation (IV+V) team.

*TRL 9* – Documentation of the information technology completed, approved and issued. Operational limits of the software are understood, documented and consistent with the operational mission requirements

## 7. EXAMPLES

We will consider two examples of IT for space systems from inception through first flight. The first example follows infusion of an existing information technology (Object Oriented Programming) into a space application. The second studies development of a totally new technology (a hypothetical autonomous planner based on heuristic principles) intended for spaceflight.

*Example 1 – Object Oriented Programming (OOP) for the Cassini Attitude and Articulation Control System (AACCS)*

OOP is based on the notion that it is possible and useful to encapsulate information and to treat the resultant information as an object. OOP then goes on to define types of properties of objects, possible manipulations of objects, and potential relationships between objects. In actuality, OOP is definition of different classes or types of knowledge, how items of knowledge can be packaged, handled, manipulated and connected to form larger knowledge entities, and handling systems.

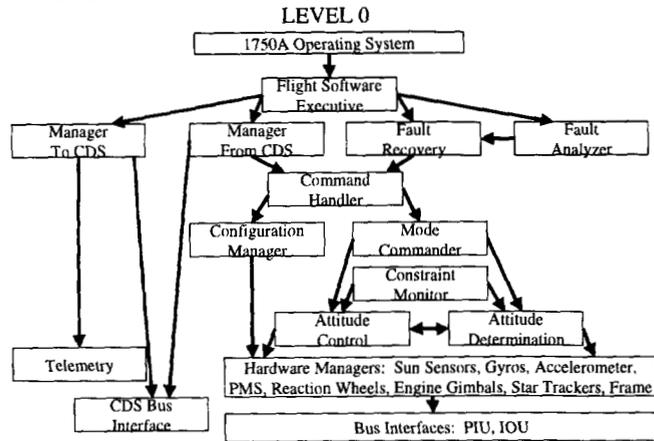
This example discusses steps taken to mature the OOP approach for the Cassini AACCS control software. Because the control software in itself does not represent new technology, we can clearly show the maturity progression of the OOP approach as applied to space flight by itself. OOP as a paradigm available to computer scientists in general was already matured at the start of this example, but software designed with OOP had not been space-qualified.

*TRL 1* – OOP formally identified as an off-the-shelf approach that might be useful for control software. The team produced documentation including a definition of the concept of OOP as applied to generic, COTS, non-flight software, and a qualitative estimate of the expected benefit.

*TRL 2* – The AACCS team produced a high-level breakdown of the software functions into OOP principles, as illustrated in Figure 2 below. The team documented its rough approach and determined that no obvious incompatibilities existed. The team identified the specific performance requirements of the software (viz. relevant environment), including target processor, language, operating system, timing requirements, and interfaces. The team also quantified the expected benefits of the approach.

*TRL 3* – The team produced portions of the AACCS software coded in the OOP paradigm. These portions were sufficient to test the end-to-end data flow and check critical functions. The results of these tests were documented to prove feasibility of the software approach.

## CASSINI AACCS FLIGHT SOFTWARE ARCHITECTURE



**Figure 2:** Cassini AACCS Software Breakdown (after Hackney, et. al. [2])

An example critical function is given below in Fig. 3. These functions were not yet integrated into a completed prototype, but were collectively defined at a level sufficient to demonstrate all critical functions of the ACS software.

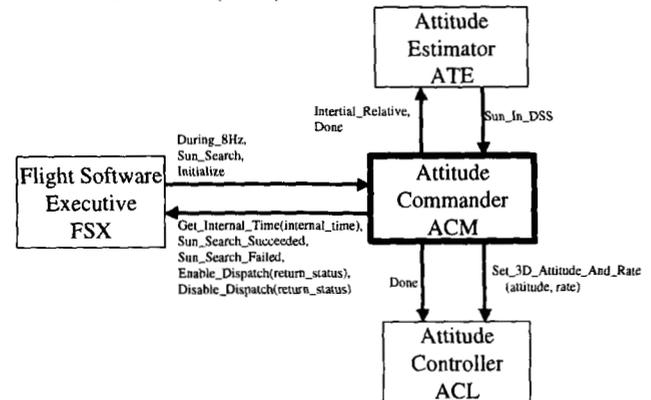
*TRL 4* – The team produced a prototype incorporating all AACCS software functions. This prototype was tested in a laboratory environment, using non-flight computers, simplified interfaces, and without strict timing requirements. This environment was the Flight Software Development System (FSDS). The test and its results were documented. These tests provided sufficient information for a defensible performance estimate of the software, allowing prediction of its performance in the relevant environment.

*TRL 5* – The AACCS software was tested in the relevant environment. This implies running in the correct language and on the correct operating system, using actual flight software interfaces and drivers to hardware and other flight software (such as the Command and Data Subsystem, or CDS), and with strict timing requirements. The test was conducted using a processor and computing hardware equivalent to flight articles. These tests included “stressful cases” designed to measure software performance at the corners of the operating envelope.

*TRL 6* – Software was tested in a complete system simulation. This includes the flight processor and operating system, and all hardware-in-the-loop. This facility was the Integration and Test Laboratory (ITL). The test program and its results were documented.

*TRL 7* – Software is tested in space in the early days of the Cassini mission.

## Object Diagram Attitude Commander (ACM)



**Figure 3:** Example critical function prototype (after Hackney, et. al. [2])

### Example 2 – Development of a Heuristic Planner

This example considers development stages of a typical mission planner. Planners are autonomy components that are responsible for generating sequences of events for spacecraft to follow in order to achieve certain goals. We will consider a planner based on entirely heuristic principles.

*TRL 1* – Observed the principle that well-crafted heuristics can be used for a more efficient planner. Documented the basic approach and made a qualitative assessment of its benefit.

*TRL 2* – Defined structure of a planner for spacecraft management. Identified the relevant environment (e.g. VxWorks, RAD750 processor, specific memory and timing targets, or at least the notion of a real-time operating system (RTOS), a reduced instruction set computer (RISC) and generalized performance limits on expected space qualified hardware to be available at the earliest envisioned date of mission insertion) and ascertained there were no intrinsic incompatibilities between the environment and the technology. Quantified benefits to cheaper operations, increased science data return, etc. given specified assumptions about the spacecraft.

It is important to note that environmental requirements should be defined early in development, here at TRL 2, in order to retard “creeping” development costs as the technology matures. It is all too easy to relax the relevant environment in response to difficulties or uncertainties in technology development. While the relevant environment will be clarified with respect to *interactions* on-board the spacecraft (at TRL 5, after a laboratory prototype has been validated), the *IT execution* environment must be fully specified at TRL 2.

*TRL 3* – Constructed a preliminary heuristic temporal database. Completed key elements of an inference engine operating on the database. End-to-end experiments conducted on partially populated database successfully generate specific plans. The scope and results of these tests are documented.

At *TRL 3*, both the method and the underlying IT components need not be fully developed – elements of the process may be done “by hand,” and that includes the development environment. The technology provider may meet the IT needs through handmade retrofits of existing tools, such as using an off-the-shelf C language compiler and rough application-specific subroutines to simulate the particular development environment needed for the planner. The technology provider does not need to create the required development environment at this *TRL*. However, the finished product and development process must be repeatable, and the provider must document the functional requirements of the proposed IT development environment. Therefore, at *TRL 3*, both the IT execution and development environments must be fully specified.

*TRL 4* – Laboratory environment prototype completed. Prototype tested with a realistic segment of mission scenario and converged upon a plan result meeting all constraints within the predicted time. The test cases may be non-stressing at this time but must be realistic. These tests should provide the basis for a performance estimate sufficiently detailed to predict planner performance in the relevant environment. The performance prediction method, model, simulation or equations is documented and comprises a performance prediction model.

At *TRL 4*, the underlying IT development environment required for the finished product must exist, but only at a level of coverage and refinement sufficient to complete the breadboard for this specific technology. Optimization, expansion, and population of the development environment sufficient to broadly apply the underlying IT is not required until after the first flight of this specific product (*TRL 7*).

*TRL 5* – Advanced prototype completed and tested in the relevant environment. The relevant environment is refined to accurately represent interactions with other spacecraft subsystems, and these interactions are documented. This prototype planner is stressed with difficult scenarios, including a wide variety of boundary conditions and subtly ambiguous test cases, and is demonstrated to produce correct and predictable results in a predictable length of time. The tests include a full simulation of the flight environment and simulated interacting components with equivalent interfaces. The testbed imposes realistic memory and timing requirements and is conducted on an equivalent flight processor. Results of these tests are documented and the predictive performance model is updated and validated for the relevant environment.

At *TRL 5* we have finalized the environment specification of the technology, including the IT environment (from *TRL 2*), the development environment (from *TRL 3*), and lastly the spacecraft environment as pertains to interactions with spacecraft subsystems (*TRL 5*).

*TRL 6* – Full brassboard simulation of the planner, using flight processors and computing environment, and all interfaces to other systems is tested and verified. Results of these tests are documented and shown to be consistent with expectations from the prototype and the predictive performance model.

*TRL 7* – First flight of the planner technology, possibly in a shadow-mode or end-of-mission experiment on board the host vehicle. Operation in actual environment, under real conditions for a significant length of time meets performance model predictions.

## 8. CONCLUSION

This paper outlines a new standard for assessment of spacecraft information technologies, historically a difficult task to manage. As information technologies mature and become increasingly complex in spacecraft, NASA and the New Millennium Program must concentrate on management and development of such technologies for space. Our modified system of Technology Readiness Levels sets concrete gates for developers of information technologies, providing guidance above and beyond the standard *TRL* definitions.

As we have seen, the principal difference is in the question of environment. Information technologies must be tested in the relevant information environment on the path to spaceflight, representing computer processors, operating systems, interfaces, and control issues typically found in spacecraft. This is a considerable departure from the physical space environment used to test ordinary technologies. Nonetheless, with careful consideration the path of information technology development can be handled in a similar fashion.

## REFERENCES

- [1] J. Mankins, “Technology Readiness Levels,” NASA White Paper, April 1995.
- [2] J. Hackney, D. Bernard, R. Rasmussen, “The Cassini Spacecraft: Object Oriented Flight Control Software,” *Proceedings of the 16<sup>th</sup> Annual American Astronautical Society Guidance and Control Conference*, February 6-10, 1993.

**Ryan Mackey** received his B.A. degree from the University of California at Santa Cruz (1993) for Mathematics and Physics, and went on to an M.S. (1994) and Eng. (1997) in Aeronautics at Caltech. He is presently a senior researcher and charter member of the Ultracomputing Technologies Research Group at the Jet Propulsion Laboratory. His research centers upon revolutionary computing methods and technologies for advanced machine autonomy, specifically deep space missions, UAVs and maintainable aerospace vehicles. His interests also include quantum- and biologically-inspired computing.

**Rafi Some** has been in the aerospace and computing industry for 27 years, with the last five at the Jet Propulsion Laboratory. He has worked to develop varied computing systems and technologies, including spacecraft avionics. He received his B.S. in Electrical Engineering from Rutgers University. His current duties include staff technologist for NASA's New Millennium Program.

**Abdullah Aljabri** supervises the Autonomy Software Technology Infusion Group at the Jet Propulsion Laboratory, and also serves as staff technologist for NASA's New Millennium Program. He received a B. Tech in Aeronautical Engineering and Design from the Loughborough University of Technology and M.S. in Aerospace Engineering from Penn State. His efforts include the Remote Agent, first flown on the DS-1 Spacecraft, and implementation of advanced computing methods in space borne systems.