NASA/TM—2013-216595

Space Telecommunications Radio System (STRS) Application Repository Design and Analysis

Louis M. Handler
Glenn Research Center, Cleveland, Ohio

November 2013

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question to *help@sti.nasa.gov*

- Fax your question to the NASA STI Information Desk at 443–757–5803

- Phone the NASA STI Information Desk at 443–757–5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2013-216595

Space Telecommunications Radio System (STRS) Application Repository Design and Analysis

Louis M. Handler
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

November 2013

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Preface

This document describes the STRS application repository. STRS is an architecture standard for NASA space communication radio transceivers. This architecture is a standard for communication transceiver developments among NASA missions. Although the architecture was defined to support space-based platforms, the architecture may also be applied to ground station radios.

The STRS Architecture strives to provide commonality among NASA radio developments to take full advantage of emerging software defined radio technologies from mission to mission. This architecture serves as an overall framework for the design, development, operation, and upgrade of these software based radios.

The STRS application repository is the vehicle for transmission of technology from mission to mission. The STRS application repository will support software and firmware porting and reuse.

This document is under the configuration management of the Glenn Research Center (GRC) at Lewis Field. Change requests and comments to this document shall be submitted to the contact below along with supportive material justifying the proposed change.

Questions and proposed changes concerning this document shall be addressed to:

STRS Architecture Manager
Communications Division
Glenn Research Center
Mail Stop 54-1
Cleveland, Ohio 44135

or email contact to:

STRS@lists.nasa.gov

Contents

List of Tables

List of Figures

Space Telecommunications Radio System (STRS) Application Repository Design and Analysis

Louis M. Handler
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

1.0 Executive Summary

This document describes the Space Telecommunications Radio System (STRS) application repository for software-defined radio (SDR) applications intended to be compliant to the *STRS Architecture Standard*. STRS is an open architecture for NASA space and ground radios. The STRS standard provides a common, consistent framework to develop, qualify, operate and maintain complex reconfigurable and reprogrammable radio systems.

The STRS application repository is intended to capture knowledge, documents, and other artifacts for each waveform application or other application outside of its project so that when the project ends, the knowledge is retained. The STRS application repository is the vehicle for transmission of technology from mission to mission. The STRS application repository will aid in the make, buy, or reuse decision. The feedback and lessons learned are used for continuous improvement; that is, to improve the waveform application, the STRS application repository, and the *STRS Architecture Standard*. The STRS application repository supports software and firmware porting and reuse. The STRS application repository design supports NASA Procedural Requirements (NPRs) for performing software engineering projects and NASA's release process.

The intended audience of this *STRS Application Repository Design and Analysis* document is the software, firmware, and hardware developers of the waveform applications, to provide information about the submission of artifacts to the STRS application repository, to provide information to the potential users of that information, and for the systems engineer to understand the requirements, concepts, and approach to the STRS application repository.

The *STRS Architecture Standard* encourages the development of waveform firmware and software that are modular, portable, reconfigurable, and reusable. To facilitate reuse of the software and firmware, sufficient information about the hardware, hardware interfaces, firmware interfaces, and configuration must be kept. Potential users must be able to search for an application that might work for their new requirements. NASA's release process must be followed to ensure that the legal procedures are followed. If allowed, the requested software, firmware, and documentation would be released to the new user for further evaluation and test.

2.0 Introduction

The *STRS Architecture Standard* defines an open architecture specification for NASA space radios and is part of the larger STRS program currently underway to define NASA's application of software defined, reconfigurable technology to meet future space communications and navigation system needs. Software-based reconfigurable transceivers (RTs) and SDRs enable advanced operations potentially reducing mission life cycle costs for space platforms. The objective of the open architecture for NASA space SDRs is to provide a consistent and extensible environment on which to construct and operate NASA waveforms for space applications, targeting radio designers and developers. The open architecture provides a framework for developing the radios and leveraging earlier efforts by reusing various components of the architecture developed in other NASA programs.

An open architecture enables cost reduction in system development and operations by promoting and enabling multiple vendor solutions and interoperability between independent hardware and software technologies. The architecture supports existing (e.g., legacy) communications needs and capabilities, while providing a path to more capable, advanced waveform development and mission concepts.

A key concept enabled by the architecture specification is reuse of previously developed hardware and software components. The ability to reuse components is accomplished by defining the various hardware and software interfaces, and providing additional layers to the architecture to abstract the software from the platform hardware. By consistently specifying these interfaces and publishing them as part of the architecture specification, the various modules can be replaced and updated with a minimum amount of changes, since the interface is specified and rules are provided for each component.

The purpose of the STRS application repository is to store the original artifacts from which each STRS application or service may be recreated, reused, or ported to another system. The storage of such artifacts will be maintained by a configuration management system that keeps track of any changes. The artifacts would likely include platform independent models, platform-specific models, source code, scripts, makefiles, configuration files, and documentation. The documentation must include a description of the application purpose and design, the process to recreate the application, systems and tools used, and application execution.

2.1 Terminology

Software defined radio is a relatively new technology area, and industry consensus on terminology is not always consistent. Some of the confusion exists when the various organizations and standards bodies define different radio terms associated with the actual amount of reconfigurability of the radio. Since the radios require at least some dedicated hardware to compliment the software, the reality of today's radios is varying degrees of reconfigurability based upon the signal processing requirements and the choice of hardware components. Definitions associated with software defined radios range from legacy transceivers with software and firmware cast in digital hardware processing to the ideal software radio that digitizes the RF signal at the antenna and has all processing done in software. For purposes of the STRS architecture and architecture specification, the following key terms are repeated from the *STRS Definitions and Acronyms* document for immediate reference.

Conventional or legacy radio is defined as a nonprogrammable radio designed for one fixed configuration for producing a single waveform at a specified frequency. The radio may have limited options for tuning, data rate, etc. or may even carry multiple types of data, but is incapable of adapting to new waveforms.

Reconfigurable transceiver is defined as a radio with limited processing and selectable remote reconfiguration (e.g., filter parameters and modulations). A reconfigurable radio is a radio whose hardware functionality can be changed under software control. Reconfiguration control of such radios may involve any element of the radio communication network.

A *software defined radio* is a radio in which some or all of the physical layer functions are implemented in software and/or firmware. An SDR performs significant amounts of signal processing in a general purpose computer, or a reconfigurable digital electronic device. The design goal of reconfigurability is to produce a radio that can receive and transmit a new form of radio signaling protocol by running new software or firmware. An SDR may have its functionality defined in software, but not be reconfigurable. Given the constraints of today's technology, there is still some RF hardware involved for front end processing.[1]

[1]"Software Radio Architecture, Object-Oriented Approaches to Wireless Systems Engineering," Joseph Mitola, 2000.

A s*oftware radio* is an extension of an SDR with more functionality implemented in software running on a GPP as opposed to ASICs and FPGAs. A software radio implements communications functions primarily through software in conjunction with minimal hardware. Software radios are the ideal SDR in which digitization occurs at the antenna and the majority, if not all functions are performed in software.

The term architecture also has different terminology definitions.

System architecture is defined as an abstract description of the entities of a system, and the relationship between the entities.

The definition of *architecture* is: …a comprehensive, consistent set of functions, components, and design rules according to which radio communications systems may be organized, designed, constructed, deployed, operated and evolved over time. A useful architecture partitions functions and components such that a) functions are assigned to components clearly and b) physical interfaces among components correspond to logical interfaces among functions. [1]

An *open* architecture is one whose functions, interfaces, components, and/or design rules are defined and published. An *open system* has characteristics that comply with specified, publicly maintained, readily available standards. Open systems architecture is nonproprietary and a key attribute is the layered hierarchical structure, configuration, or model. An open SDR architecture applied to radios provide partitioned software modules controlled by managing software (compliant with the architecture rules set) that meet defined published interfaces (e.g., API's) to allow software portability and scalability across hardware platforms.

The architecture terms defined above are general, and there are more specific definitions for the software defined radio case. *Reconfigurability* is the ability to modify functionality of a radio by changing the operational parameters without requiring a software update. An *application* is an executable software program that may contain one or more software modules. The executable software exhibits predetermined functionality. A primary example of an STRS application is the waveform application. An STRS application must comply with the STRS architecture. An STRS application is executable software and/or firmware that is abstracted from the radio platform. The software and firmware modules and components are reuseable and portable. A waveform comprises the end to end functionality from the data input to the radiated signal and from the received signal to the data output. *Services* are software programs running on the software radio that provide functionality available for use by other applications. Waveforms and services are types of STRS applications. An API is a formalized set of software calls and routines that can be referenced by the application program to access supporting system or network services.

2.2 Roles and Responsibilities

As described in the *STRS Architecture Standard*, the *STRS application developer* creates the application, performs unit tests, and documents the functionality. The *STRS application integrator* creates the predeployed configuration file in XML 1.0 and transforms it to the deployed configuration file, which he uses to instantiate and test the application. There may be a different *STRS application integrator* for each STRS application. The *platform provider* is responsible for the sample application and sample configuration files and takes the role of *STRS application integrator* for them.

The STRS repository manager is a member of NASA's STRS project and is responsible for the security, confidentiality, integrity, availability, authenticity, accountability, and assurance of the STRS application artifacts submitted to the STRS application repository for possible future reuse, either wholly or in part. The STRS repository manager manages requests for access to artifacts in the repository with the aid of NASA's Software Release Authority (SRA). The STRS repository manager protects the artifacts from unauthorized disclosure, access, use, modification, or destruction. The STRS repository manager is also responsible for maintaining an index of what STRS applications and artifacts are available and their properties.

The STRS liaison is a member of NASA's STRS project who answers STRS questions and advises the project manager about the STRS requirements, STRS application repository, and STRS compliance, as well as providing STRS training and documents. The STRS liaison collects lessons learned for inclusion in the STRS application repository and for consideration as improvements to *STRS Architecture Standard*. The STRS liaison informs the STRS repository manager of current availability of artifacts and the readiness for STRS compliance testing. The STRS liaison role may be subdivided so as to assign different parts to different individuals.

Any organization may define other supporting roles and requirements as necessary to facilitate its tasks. For example, NASA mission and project management may add supporting roles and requirements as specified in NASA standards and NASA procedural requirements (e.g., NPR 7150.2). One might define project roles for project management support, configuration management, legal review, document review, code review, change control board, testing, and operations. Each NASA mission or project requiring STRS architecture should contact the STRS team to obtain STRS training, get questions answered, and obtain personnel to participate in reviews. Each NASA mission or project requiring STRS architecture should provide appropriate personnel to ensure that STRS is complied with and noncompliances from STRS compliance testing on both documentation and software are addressed.

2.3 Scope

The *STRS Application Repository Design and Analysis* document is divided into several different sections, organized as follows: Section 1.0 provides the executive summary. Section 2.0 introduces the purpose of this report, the terminology used, and the scope of materials contained within the architecture specification. Section 3.0 lists the applicable documents, which is an important section, since a number of documents are assumed as background material.

Requirements are stated within the subsection where they apply. They are not stated in order of importance. Section 4.0 describes the background of the STRS application repository and the basis for the requirements. It is divided into the purpose for NASA and how NASA procedure requirements (NPRs) raise additional requirements for release. The STRS application repository artifact requirements from the *STRS Architecture Standard* are restated in Section 5.0 as well as a discussion of additional reuse requirements.

The project life cycle is developed in Section 6.0. The STRS application repository metadata, including index and metrics, is described in Section 7.0. The general use case for the STRS application repository is described in Section 8.0.

Criteria for estimating an overall reuse readiness level (RRL) as well as individual reuse readiness levels in nine categories are provided in Section 9.0. Section 10.0 shows the information required by the form NF-1679, Disclosure of Invention and New Technology. Section 11.0 shows the information required by the NASA Software Release Request Authorization (SRRA) form. Section 12.0 provides criteria for estimating a technology readiness level (TRL).

3.0 Documents

3.1 Applicable Documents

1. Space Telecommunications Radio System (STRS) Architecture Standard, December 2010, TM 2010-216809, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110002806_2011001718.pdf
2. NPR 2210.1, Release of NASA Software, http://nodis3.gsfc.nasa.gov/npg_img/N_PR_2210_001C_/N_PR_2210_001C_.pdf

3. GLPR 2210.1, Software Release, https://knowledgeshare.grc.nasa.gov/eRoomReq/Files/NASAc1f1/GRCKnowledgeBase/0_eff3/GLPR%202210.1A.pdf
4. NF-1679, Disclosure of Invention and New Technology, https://ntr.ndc.nasa.gov/nf1679.pdf
5. NASA Software Release Request Authorization (SRRA) form obtained from Software Release Authority
6. NPR 7120.8, NASA Research and Technology Program and Project Management Requirements, http://nodis3.gsfc.nasa.gov/npg_img/N_PR_7120_0008_/N_PR_7120_0008_.pdf
7. NPR 7150.2, NASA Software Engineering Requirements, http://nodis3.gsfc.nasa.gov/npg_img/N_PR_7150_002A_/N_PR_7150_002A_.pdf
8. NASA-STD-8719.13, Software Safety Standard, http://www.hq.nasa.gov/office/codeq/doctree/871913B.pdf
9. NASA-STD-8739.8, Software Assurance Standard, http://www.hq.nasa.gov/office/codeq/doctree/87398.pdf

3.2 Reference Documents

1. Space Telecommunications Radio System (STRS) Definitions and Acronyms, May 2008, TM 2008-215445, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20090005977_2009004914.pdf
2. Space Telecommunications Radio System (STRS) Architecture Goals/Objectives and Level 1 Requirements Document, June 2007, TM 2007-215042, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008862_2008008550.pdf
3. Space Telecommunications Radio System (STRS) Compliance Testing, December 2011, TM 2011-217266,
4. http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120000912_2012000901.pdf
5. Reuse Readiness Levels (RRLs), April 2010, by NASA Earth Science Data Systems – Software Reuse Working Group,
6. http://www.esdswg.org/softwarereuse/Resources/rrls/RRLs_v1.0.pdf

3.3 Background Documents

1. JTRS Information Repository (IR) Access Request Review / Approval Process, http://jtrs.spawar.navy.mil/account/forms/IR_Access_Request_Review_and_Approval_Process.pdf
2. Section 508 of the Rehabilitation Act (29 U.S.C. '794 d), as amended by the Workforce Investment Act of 1998 (P.L. 105 - 220), August 7, 1998

4.0 STRS Application Repository Requirements Development

4.1 STRS Application Repository Purpose

The purpose of the STRS application repository is to reduce the cost of developing software defined radios by:

- Shortening time to deployment
- Promoting porting and reuse of previously-developed artifacts
- Reducing nonrecurring engineering costs
- Reducing duplication of effort
- Capturing knowledge of what was done before
- Improving quality when reusing previously tested artifacts

Furthermore, since projects have a finite lifetime, the relevant waveform knowledge should be captured outside of the project or mission. The knowledge should include

- Design
- User documentation
- Model data
- Tool description
- Operating environment (OE) description
- Platform-specific wrapper
- Hardware interface description (HID)
- Source files
- Configuration file information
- Test plan and test results

The saved information provides a starting point for succeeding SDR application development. Metadata aids a potential user in the preliminary decision whether to make, buy, or reuse an SDR application. Generally, each SDR will have a unique OE corresponding to the specific hardware, the OE is proprietary, and therefore is not kept as part of the STRS repository. However, enough platform information, for both OE and hardware, must be kept so that the SDR application functionality is clear.

To shorten time to deployment, preapproved application software is desirable. As described in the next section, NPR 2210.1 and NPR 7150.2, require the NPR 7150.2 classification and corresponding documentation so that each dependent project can show that the software was designed, implemented, reviewed, and tested according to plan.

Precertified application software is also desirable; however, this is not usually possible because of NASA's testing requirements for flight. But it may be possible to reuse test plans and procedures to facilitate the certification process.

4.2 STRS Application Repository Requirements Basis

In the *STRS Architecture Standard*, there is only one STRS requirement, STRS-12, that specifically addresses the need for the STRS application repository; however, it is implicit where artifacts associated with the STRS application are required to be delivered to NASA. These STRS requirements are summarized in this document in Section 5.1, STRS Requirements for the STRS Application Repository.

All STRS application repository submissions must adhere to NPR 2210.1, Release of NASA Software. NPR 2210.1 is applicable to the reporting, review, assessment, and release of all software produced by or for NASA except for bug fixes or classified as top secret, secret, or confidential. According to NPR 2210.1 section 1.8, the responsible center office or project must recommend a release category under NPR 2210.1 section A.2: Public Release, Open Source Release, U.S. and Foreign Release, U.S. Release Only, and U.S. Government Purpose Release where the latter release category includes the following five subcategories: Beta Release, Project Release, Developmental Release, Interagency Release, NASA Release. Unrestricted release without an appropriate software use agreement (SUA) or release record is not allowed by Agency policy.

According to NPR 2210.1 section 1.8, the responsible center office or project must report the software to be released to the Software Release Authority (SRA), currently Jason Hanna. The information reported must include:

1. Any programmatic restrictions on release of the software

2. The software classification (A-H) as defined in NPR 7150.2, NASA Software Engineering Requirements
3. Whether the software complies with the software engineering and assurance requirements of NPR 7150.2, NASA Software Engineering Requirements, and NASA-STD-8739.8, Software Assurance Standard, for the applicable software classification
4. Whether the software is safety-critical software as defined in NASA-STD-8739.8, Appendix A, The Software Assurance Classification Assessment, and if so, whether it complies with the software safety requirements of NASA-STD-8719.13, Software Safety Standard
5. The software's Technology Readiness Level (TRL) as defined in NPR 7120.8, NASA Research and Technology Program and Project Management Requirements (and reproduced in Appendix E of NPR 2210.1)
6. Any software documentation, as defined in paragraph NPR 2210.1 section A.1.14, that is proposed (or available) for release with the software (e.g., owner's manuals, user's manuals, installation instructions, operating instructions)
7. Whether any known export restrictions apply to the software
8. Whether the software includes any Open Source or other third party software
9. Whether Open Source Release of the software is proposed
10. Whether the software includes any embedded computer databases
11. Whether the software is Section 508 compliant

This information to be reported to the SRA is encapsulated in two forms:

1. NF-1679, Disclosure of Invention and New Technology form. See Section 10.0 for an outline of the NF-1679 form.
2. NASA Software Release Request Authorization (SRRA) form. See Section 11.0 for an outline of a recent version of the SRRA form.

These forms also require a requirements mapping matrix to be filled in based on NPR 7150.2, Appendix D, Requirements Mapping Matrix. Furthermore NPR 7150.2, SWE-027, requires sufficient documentation to proceed with reuse by documenting:

1. Requirements
2. Usage instructions
3. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights
4. Support plan
5. Verification and Validation (V&V) test plan

To aid in the make/buy/port/reuse decision, the following additional requirements for metrics must be added:

1. SLOC—source lines of code
2. Time and level of effort to make, port, or reuse with "from/to" information (or estimate)
3. RRL—reuse readiness level (see Section 9.0)

4. TRL—technology readiness level (see Section 12.0)
5. STRS compliance testing results
6. Any waivers and/or deviations
7. Lessons learned
8. History of attempted reuse whether fully or partially reused, or considered and later rejected

Some suppliers consider item 2 to be competition-sensitive and will be unwilling to provide this information with the application. Therefore, an alternative has been proposed of supplying a standardized estimate using a software cost estimation tool such as COCOMO II. Information on COCOMO can be found at http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html.

In addition, the STRS application repository must have a searchable index to find applications by:

1. Functional description
2. Same/similar functionality
3. Originating from same application after previous port/reuse
4. Running on same platform
5. Running on same OE
6. Originating from same innovator

The index must be searchable by projects interesting in using items in the repository. At least a portion of the index including the functional description must be searchable without restriction as to the origin of the request so that companies or projects may determine an initial level of interest before beginning NASA's release process.

The functional description may be created from the NF-1679 data for the descriptive title (1), abstract (9), description of the problem (Section I), and purpose (Section II). Whether functionality is the same, similar, or different from any other entry in the STRS application repository may be difficult to determine from the functional description because different words may be used to describe the same thing; however, NF-1679, Section III, item F, describes reuse of pre-existing code.

5.0 STRS Application Repository Requirements

The STRS artifacts to be delivered relative to the STRS application repository are enumerated in the following section. Then, following that, additional requirements for reuse are listed.

5.1 STRS Requirements for the STRS Application Repository

The STRS requirements are fully described in the *STRS Architecture Standard*. Some platform documentation must be included to be able to understand the workings of the STRS application running on its platform.

An STRS platform must be delivered with a Hardware Interface Description (HID), which abstracts and defines the module functionality and performance as well as physically identifying how modules are integrated on a platform. The HID specifies the characteristics of each reconfigurable device, including such items as device type, processing capability, clock speeds, reconfigurability capacity, the electrical interfaces, connector requirements, and physical requirements for the delivered radio, noting any constraints. Portions of the HID could overlap with other documents, such as an interface control

document (ICD) or waveform application developers' guide describing how the STRS application controls the hardware and what the hardware does in response. In those cases, the documents could be considered part of the HID and would not have to be rewritten.

The *STRS Architecture Standard* specifies that platform providers must provide a field programmable gate array (FPGA) platform-specific wrapper, which abstracts the platform from the waveform application. Waveform applications developed on the special-purpose processor (SPM) are to be controlled with commands from the general-purpose processor (GPP) through the platform-specific wrapper.

The Hardware Abstraction Layer (HAL) is the library of functions that provides a software view of the specialized hardware by abstracting the physical hardware interfaces. The HAL API must be published so that software and firmware running on the platform's specialized hardware may integrate with the STRS infrastructure made by a different company. The HAL is not required to be stored in the STRS application repository.

STRS application configuration files are developed for every STRS application. They must contain application specific information for the installation and customization of applications. An STRS application configuration file contains specific information that 1) allows STRS to instantiate the application; 2) provides default configuration values; and 3) provides connection references to ports and services needed by the application. The information contained in the STRS configuration files will be stored in the STRS application repository so that the initial/default configuration of the STRS application is known. This information should include the XML files and XML schema as well as documentation of those.

The STRS requirements from the *STRS Architecture Standard*, pertaining to the STRS application repository, are:

(STRS-4) The STRS platform provider **shall** describe, in the HID document, the behavior and capability of each major functional device or resource available for use by waveforms, services, or other applications (e.g., FPGA, GPP, DSP, memory), noting any operational limitations.

(STRS-5) The STRS platform provider **shall** describe, in the HID document, the reconfigurability behavior and capability of each reconfigurable component.

(STRS-6) The STRS platform provider **shall** describe, in the HID document, the behavior and performance of the RF modular component(s).

(STRS-7) The STRS platform provider **shall** describe, in the HID document, the interfaces that are provided to and from each modular component of the radio platform.

(STRS-8) The STRS platform provider **shall** describe, in the HID document, the control, telemetry, and data mechanisms of each modular component (i.e., how to program or control each modular component of the platform, and how to use or access each device or software component, noting any proprietary aspects).

(STRS-9) The STRS platform provider **shall** describe, in the HID document, the behavior and performance of any power supply or power converter modular component(s).

(STRS-12) Application development artifacts **shall** be submitted to the NASA STRS Repository. The use will be subject to the appropriate license agreements. The application development artifacts **shall** include the following:

1. High-level system or component software model
2. Documentation of application firmware external interfaces (e.g., signal names, descriptions, polarity, format, data type, and timing constraints)

3. Documentation of STRS application behavior
4. Application function sources (e.g., C, C++, header files, VHDL, and Verilog)
5. Application libraries, if applicable (e.g., EDIF and DLL)
6. Documentation of application development environment and tool suite
    a. Include application name, purpose, developer, version, and configuration specifics
    b. Include the hardware on which the application is executed, its OS, OS developer, OS version, and OS configuration specifics
7. Test plan and results documentation
8. Identification of Flight Software Development Standards used
9. Version of the STRS Architecture Standard used

(STRS-14) The STRS SPM developer **shall** provide a platform-specific wrapper for each user-programmable FPGA on the SPM, which performs the following functions:

1. Provides an interface for command and data from the GPM to the waveform application
2. Provides the platform-specific pinout for the application developer. This may be a complete abstraction of the actual FPGA pinouts with only waveform application signal names provided.

(STRS-15) The STRS SPM developer **shall** provide documentation on the firmware interfaces of the platform-specific wrapper for each user-programmable FPGA on the SPM, which describe the following:

1. Signal names and descriptions
2. Signal polarity, format and data type
3. Signal direction
4. Signal timing constraints
5. Clock generation and synchronization methods
6. Signal registering methods
7. Identification of development tool set used
8. Any included noninterface functionality

(STRS-99) The STRS application developer **shall** document the necessary application information to develop a predeployed application configuration file in XML 1.0.

(STRS-100) The STRS application integrator **shall** provide a predeployed application configuration file in XML 1.0.

(STRS-101) The predeployed STRS application configuration file **shall** identify the following application attributes and default values:

1. Identification
    a. Unique STRS handle name for the application
    b. Class name (if applicable)
2. State after processing the configuration file
3. Any resources to be loaded separately

a. Filename of loadable image

b. Target on which to put loadable image file

c. Target memory in bytes, number of gates, or logic elements

4. Initial or default values for all distinct operationally configurable parameters

(STRS-102) The STRS platform provider **shall** provide an XML 1.0 Schema Definition (XSD) file to validate the format and data for predeployed STRS application configuration files, including the order of the tags, the number of occurrences of each tag, and the values or attributes.

(STRS-103) The STRS platform provider **shall** document the transformation (if any) from a predeployed application configuration file in XML into a deployed application configuration file and provide the tools to perform such transformation.

(STRS-104) The STRS application integrator **shall** provide deployed STRS application configuration file for the STRS infrastructure to place the STRS application in the specified state.

5.2 Additional Requirements for the STRS Application Repository

The STRS application repository must enable reuse and porting of the artifacts. This implies many additional requirements. Most of the information stored will require the STRS repository manager to obtain or create the necessary information. If documents are already created, they should not be rewritten. Informally, the requirements would be:

1. A searchable index must be created to describe each set of artifacts, their purpose, deployment date, and availability for reuse.

   Rationale: a repository is only helpful if knowledge of its contents is available.

2. The artifacts themselves must be protected from inadvertent access, modification, or reuse by restricted entities.

   a. The STRS application repository may include agreements, contracts, copyrights, etc.

   b. The STRS application repository must be configuration managed and backed up.

   c. The STRS application repository must be protected with access codes/passwords.

   d. The security of the artifacts must be addressed concerning International Traffic in Arms Regulations (ITAR) and Export Administration Regulations (EAR).

   Rationale: a repository is useful if its contents may be trusted to match its description, not change after each use except for lessons learned, and be accessed only by those with a need to know. It has been noted that various subsets of artifacts may have different restrictions on the access and reuse of the information.

3. The artifacts must contain sufficient documentation to proceed with reuse according to NPR 7150.2A.

   Rationale: NPR 7150.2A requires specific documentation for each software classification when the software is reused according to SWE-027:

   a. The requirements that are to be met by the software component are identified.

   b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).

c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed.

d. Future support for the software product is planned. (Note that, although future support must be addressed, NASA may or may not choose to pay for it, so that it could be documented as YES, NO, NA, or $$$.)

e. The software component is verified and validated to the same level of confidence as would be required of the developed software component.

4. The artifacts must contain a software classification determination according to NPR 7150.2A Appendix E, Software Classifications.

   Rationale: NPR 7150.2A requirement SWE-020, SWE-021, SWE-126, SWE-132.

5. The artifacts must contain a project compliance matrix according to NPR 7150.2A Appendix D, Requirements Mapping Matrix and its classification.

   Rationale: NPR 7150.2A requirement SWE-021, SWE-125, SWE-127, SWE-139.

6. If the primary responsibility for the project is at NASA Glenn Research Center, the information for the NASA software inventory must be reported to the software inventory point of contact. This information includes:

   a. Project number; i.e., work breakdown structure (WBS)

   b. Project or subproject title

   c. Mission directorate responsible

   d. Project point of contact (name, organization code, phone number, email)

   e. NPR 7150.2A classification

   f. NPR 7150.2A compliance matrix existence (yes, no, or draft)

   g. Primary programming and modeling language(s) (up to 3)

   h. If class C or above, additional information will be obtained from the project or subproject point of contact.

   Rationale: NPR 7150.2A specifies that the NASA software inventory is maintained by the NASA office of the chief engineer. At NASA Glenn, the software inventory information is accumulated periodically through the Flight Software branch chief or delegated through a directorate software inventory point of contact.

7. The STRS repository manager must obtain or create NF-1679 Disclosure of Invention and New Technology.

   Rationale: NPR 2210.1 and GLPR 2210.1 require software for release to be reported to NASA. This reporting may be done electronically using the NASA electronic New Technology Reporting system (eNTRe) located at http://invention.nasa.gov/ (preferred method) or by filling in NASA Form (NF) 1679 "Disclosure of Invention and New Technology (Including Software)," also available at the eNTRe web site. NF-1679 is described in detail in Section 10.0, Disclosure of Invention and New Technology NF-1679. Since this form is required for release, it should be stored for future reference as well as submitted to the SRA.

8. The STRS repository manager must obtain or create the Software Release Request Form (SRRA).

   Rationale: GLPR 2210.1 requires the SRRA. The SRRA form is described in detail in Section 11.0, NASA Software Release Request Authorization. Since this form is required for release, it should be stored for future reference as well as submitted to the SRA.

9. The repository artifacts description must contain sufficient metrics to estimate the effort to port or reuse the application. Those metrics that may be most helpful are:

   a. Time to develop or port/reuse the STRS application or a COCOMO II computed estimate when time to develop is competition sensitive

   b. Lines of code

   c. Reuse readiness levels (RRL)

   d. Technology Readiness Level (TRL)

   e. Availability of models from which code may be generated

   Rationale: Since porting and reuse are where NASA expects to save money in the long run when adapting waveform applications for successive usage, NASA needs to verify this assumption using metrics.

10. The author(s) and technical contact of the STRS application should be identified for attribution, obtaining missing items, and possibly answering technical questions.

    Rationale: The attribution of the STRS application is important to identify whether it was written by NASA or some other entity. The attribution also facilitates obtaining updates as needed and any missing information.

11. An administrative contact should be identified for answering questions about licensing, nondisclosure, property rights, or distribution, and who would receive reports of problems, reuse, or attempted reuse.

    Rationale: This is usually the person who signs the contract. The contact for technical questions usually has no knowledge of reuse/release rights and has no connection to the artifacts after the project concludes. This administrative contact may capture problems and lessons learned so that they can be fixed or be part of product line planning. Note that reuse or attempted reuse would not be reported in a timely fashion when such information is competition sensitive.

12. The artifacts description must contain MD5 and SHA-1 checksums for each file.

    Rationale: Checksums are used to verify the integrity of files. At least MD5 and SHA-1 checksums should be computed as these are the most commonly used and, if both are used, the effects of their respective weaknesses can be disregarded because it is extremely unlikely that a given malicious file could simultaneously exploit the weaknesses of both.

6.0 Project Life Cycle Using STRS

The full life cycle of a project using STRS architecture and the STRS application repository may be illustrated as follows in Figure 6.1. In the description below of the life cycle, the STRS involvement is highlighted in bold. Also, the STRS repository involvement is double-underlined. The life cycle is based on the assumptions that the STRS Architecture is required, that the STRS application repository has been established, and that there is an STRS repository manager. The STRS application repository must be a

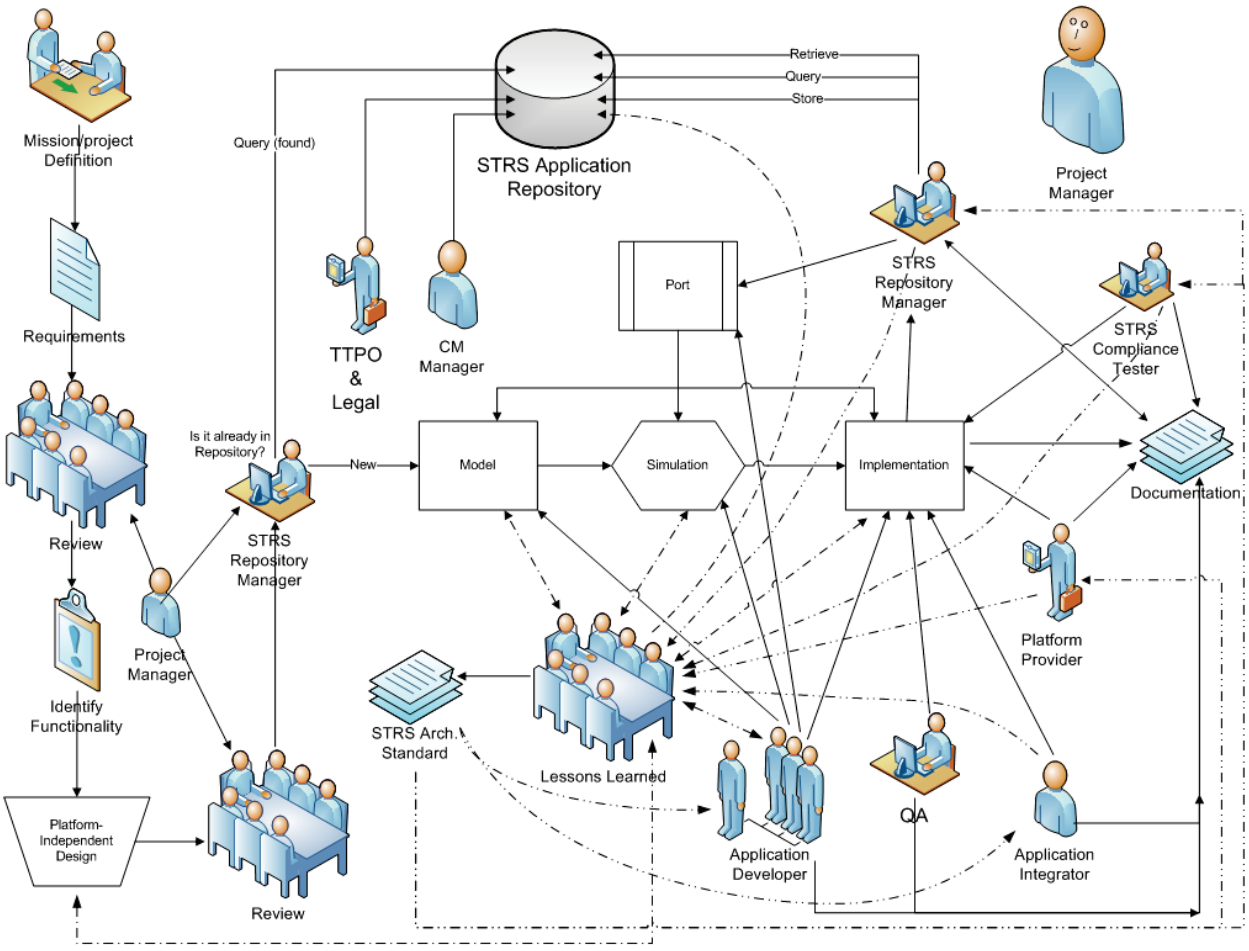

Figure 6.1.—Project Life Cycle

configuration managed area on one or more access controlled servers supporting ITAR/SBU data or the equivalent. There are considerations omitted from this life cycle that happen previous to or in parallel with it, related to concept studies, trade studies, budgeting, availability of personnel, scheduling, contracting, etc. These kinds of considerations are the responsibility of the project manager but have little to do with the STRS application repository.

1. Project begins with project manager and goals of the SDR in the system defined.
2. Analysis is performed, possibly incorporating use cases and trade studies.
3. Requirements are developed including **STRS** use. Likely areas for an **STRS** relationship are:
   a. Communication
   b. Command and data handling
   c. Telemetry
   d. Computation
   e. GPS and navigation

4. Perform requirements review. Invite **STRS** personnel participation to aid in determining any missing STRS requirements.
5. Perform NPR 7150.2A classification.
6. Identify waveform application functionality including that necessary to support ***STRS*** *Architecture Standard.*
7. Create preliminary design.
8. Perform design review. Invite **STRS** personnel participation.
9. Establish reuse, make or buy criteria (i.e., decision strategy). Decide what deliverables are needed to support project requirements, **STRS** requirements, NPR 7150.2A project compliance matrix, desired reuse readiness level (RRL), and desired technology readiness level (TRL).
10. Evaluate reuse, make, or buy.
    a. **STRS** repository manager determines whether functionality exists in **STRS** application repository.
    b. **STRS** repository manager determines reuse readiness level (RRL) for each set of artifacts implementing the required functionality and provides justification.
    c. **STRS** repository manager determines whether the artifacts in the **STRS** application repository support reuse according to NPR 7150.2A.
        (1) The **STRS** repository manager and center council determine whether proprietary, usage, ownership, warranty, licensing, disclosure, and transfer rights allow reuse. Reuse rights may be complicated so that what is put into the repository may be used in whole or part by some organizations and individuals but not others.
        (2) The **STRS** repository manager determines which artifacts may be reused; for example, requirements and design documents, models, source code, test plans, software verification and validation plans.
    d. Project manager causes time and cost estimates to be created and evaluates the options. Budgeting and scheduling should include the time and cost of the **STRS** training, supplying information to the **STRS** repository manager, and responding to **STRS** compliance testing.
11. If make or reuse:
    a. Assemble software team.
    b. Decide on development and production platforms.
    c. Acquire hardware.
        (1) Develop acceptance criteria.
        (2) Perform acceptance testing.
    d. Acquire/provide OE, HAL, HID, etc.
    e. **STRS** repository manager downloads any reusable artifacts to the development platform.
    f. Software team creates detailed design.
    g. Software team models/simulates/implements any missing software and firmware.

12. If buy:
    a. Create RFP including **STRS**, NPR 7150.2A, TRL, and RRL considerations.
    b. Create RFP evaluation criteria.
    c. Put RFP out to public.
    d. Evaluate proposals received and choose supplier.
    e. Create agreements.
    f. Acquire development and production platforms.
        (1) Develop acceptance criteria.
        (2) Perform acceptance testing.
    g. Acquire/provide OE, HAL, HID, etc.
    h. Acquire waveform application.
13. Perform **STRS** Compliance Testing and create a summary report.
14. Integrate hardware, software, and firmware into system.
15. Perform unit tests, system tests, and functional testing.
16. Feed new application artifacts and documentation to **STRS** repository manager for inclusion in **STRS** application repository.
    a. **STRS** repository manager creates cross-reference index of artifacts.
    b. **STRS** repository manager ensures that artifacts support:
        (1) **STRS** requirements
        (2) NPR 7150.2A project compliance matrix according to classification
        (3) Desired reuse readiness level (RRL)
        (4) Desired technology readiness level (TRL)
        (5) Negotiated submission and reuse agreements
    c. **STRS** repository manager captures metrics. The metrics are intended to aid in the make/buy/reuse decision as well as aid continuous improvement. Reuse readiness levels may be correlated against level of effort required (time and money).
    d. **STRS** repository manager creates searchable meta-data to support reuse evaluation.
17. **STRS** team captures lessons learned to update the ***STRS*** *Architecture Standard* and improve the reuse process, the artifacts, and the descriptions of those artifacts in the **STRS** application repository.

7.0 STRS Application Repository Data

7.1 STRS Application Repository Metadata

The STRS application repository must have a searchable index. This index would be hierarchical so that the top level contains the identification of the set of artifacts and the lower level contains the

identification of the individual artifacts. At the top level, there must be a unique identification, purpose, history, and metrics. The top level index may be enumerated as follows:

1. Unique identifier.
2. Identification of artifacts (identification) (for STRS-12)
3. Version (identification) (for STRS-12)
4. Description of overall purpose of application (identification) (for STRS-12)
5. Description of history of application
    a. Author(s) (for STRS-12)
    b. Company(s)
    c. Project
    d. Space or ground (to support TRL for NPR 2210.1)
    e. Hardware
    f. Application development environment, compilers, libraries, operating system for which application was written/used (Information to support reuse cost estimate) (for STRS-12)
    g. Start date
    h. Deployment date(s) (to support TRL)
    i. End date
    j. Entry date
6. Metrics
    a. Lines of source code (C, C++, VHDL, Verilog, etc.)
    b. Time and level of effort to develop the STRS application or equivalent
    c. Availability of models
    d. RRLs and justification (Information to support reuse cost estimate)
    e. TRL and justification (Information to support reuse cost estimate and for NPR 2210.1)
    f. Cost (Information to support reuse cost estimate)
    g. Type of legal agreement (none, GPR, etc., for NPR 2210.1)
    h. Software classification determined according to NPR 7150.2A Appendix E, Software Classifications

There must be a way to list and link the appropriate support documents and information. The supporting information could be displayed in a web page with links or in a database system.

1. Applicable legal agreements/contracts (Information to support reuse cost estimate) (for STRS-12)
2. Project compliance matrix according to classification (for NPR 7150.2A)
3. Waivers and deviations
4. NF-1679
5. SRRA

6. HID
7. Models
8. Source code
9. Standards used
10. Application libraries
11. Platform-specific wrapper for each user-programmable FPGA
12. Platform-specific documentation for OE and FPGA wrapper(s)
13. User Manual
14. STRS application behavior
15. Application development environment and tool suite
16. Test plan and results
17. STRS compliance test results
18. Initial/default configuration
19. Lessons learned

For each item, there would be meta-data containing a unique identification, purpose, history, and metrics:

1. Name (identification) (for STRS-12)
2. The filename and relative path (identification) (for STRS-12)
3. Type of item (identification) (for STRS-12)
    a. Software source
    b. Software object
    c. Firmware source
    d. Firmware configuration file (bit)
    e. Picture (e.g., jpeg, tiff, png, gif, bmp)
    f. Chart/diagram (e.g., Visio)
    g. Configuration file (e.g., xml)
    h. Word document (doc, docx)
    i. Adobe acrobat document (pdf)
    j. Html document
    k. Model
4. Purpose of item (identification) (for STRS-12, NPR 7150.2A, RRL, and TRL)
    a. Documentation: software and firmware interfaces, HAL, HID, high level system or component software model, requirements, design, user's guide, test plan, test procedures, test results, maintenance plan, safety plan, standards used, known problems, lessons learned, etc.

b. Software: waveform application, test program, application libraries, configuration file transformation, preprocessor, model, etc.

c. Firmware: source (e.g., VHDL, Verilog), FPGA configuration file (e.g., .bit), Electronic Design Automation (EDA) tool's models, Electronic Design Interchange Format (EDIF), etc.

5. Creation or processing (related information for use) (for STRS-12)
   a. Date
   b. By author(s) (for STRS-12)
   c. By some process (for STRS-12)
      (1) Description (for STRS-12)
      (2) Tool(s) (for STRS-12)
         (a) Name (for STRS-12)
         (b) Developer (for STRS-12)
         (c) Version (for STRS-12)
         (d) Configuration specifics (for STRS-12)
         (e) OS (for STRS-12)
            i. Name (for STRS-12)
            ii. Developer (for STRS-12)
            iii. Version (for STRS-12)
            iv. Configuration specifics (for STRS-12)
      (3) Input files (for STRS-12)
   d. Standards used (for STRS-12)
6. Application version description (for STRS-12)
7. Relationship to other items, dependencies
8. MD5 and SHA-1 checksums for the item

7.2 STRS Application Repository Data Files

The STRS application repository data would most likely contain the following kinds of files. The preliminary STRS application repository structure shown below is meant to be for discussion purposes only, until implementation structure is decided.

- **Metadata**
- **Source Code**
  - C/C++
  - XML Files
  - VHDL
  - DSP

    - Implementation Files
        - CORBA Interfaces
        - Waveform Component Implementations
- **Executables**
- **Models**
    - Floating-point Model
    - Fixed-point Model
    - Network Model
    - Simulation Model
    - UML Model
    - FPGA (when applicable)
    - Software Tool Environment (e.g., Simulink. Matlab)
- **Design**
    - System Sequence
    - Class
    - Use Case
    - State
    - Collaboration
- **Platform-Specific Software**
    - OE dynamically linked libraries (DLLs)
- **Platform-Specific Firmware**
    - Platform-specific wrapper
- **Documentation**
    - Description of waveform
        - Modulation
        - Compliance
        - Vendor and Contact Information
    - Platform Related
        - Vendor and Contact Information
    - Operational Environment
        - Operating System
        - Command system
        - Power system

- Software Tools Inventory
  - List of all software used to create, edit waveforms, docs, etc.
- Specifications
  - Implementation
  - Functional
  - Technical
  - Performance
- Requirements
  - Project
  - Design
- Tests
  - Plans
  - Procedures
  - Results

- **Interface Control Documents**
- **Build-Script Files**
- **Configuration files**
- **Make files/project files**
- **Diagrams**
- **Presentations**
- **User Guides/Manuals**
- **Lessons Learned**

8.0 STRS Application Repository Use Cases

The STRS application repository is used by two groups:

1. The project that creates the artifacts
2. The project that uses/ports the artifacts

These may be broken out as shown in the following Figure 8.1, with the creation of artifacts on the left and usage of artifacts on the right. In the case when NASA partners with an external organization but the project management resides within NASA, there may be no difference or only slight adjustments. If an external organization is involved, additional agreements may need to be written and signed to ensure compliance with any STRS requirements or STRS repository release restrictions.

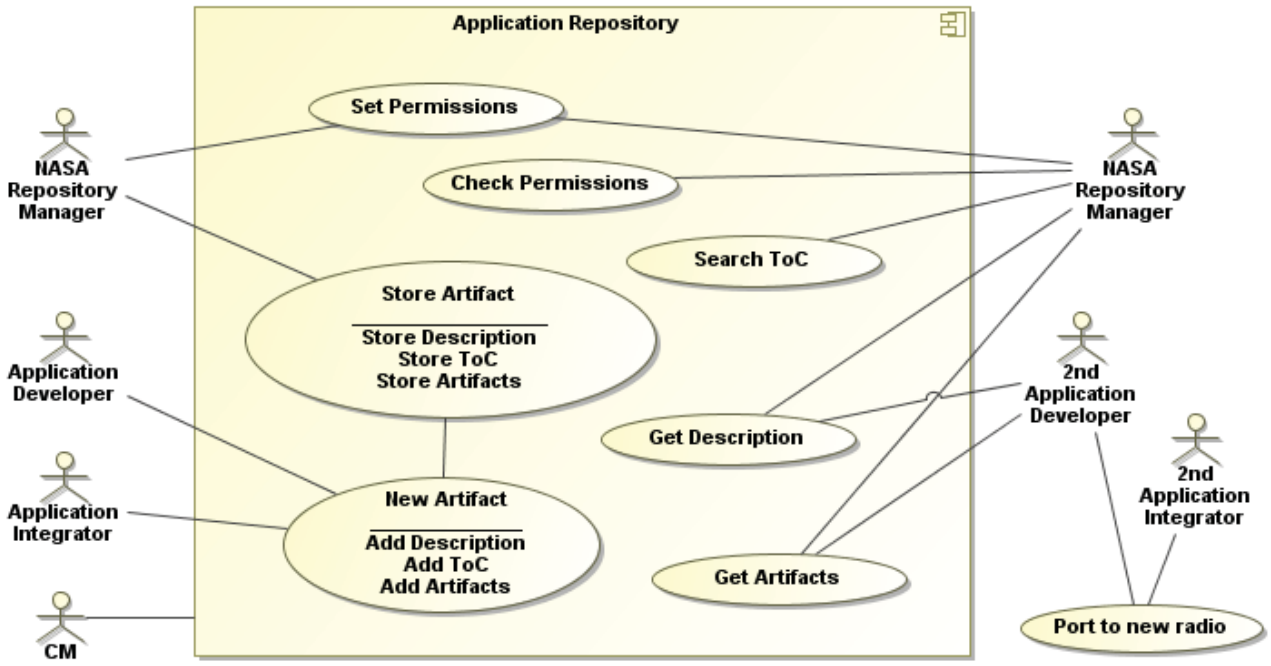

Figure 8.1.—STRS Application Repository Use Case

8.1 Push Use Case

1. The STRS application developer and application integrator create the artifacts to be put into the repository.
2. The STRS application developer and application integrator inform the project manager and STRS liaison that the project is substantially complete and artifacts are ready.
3. STRS liaison informs the STRS repository manager that artifacts are available for the STRS application repository.
4. STRS repository manager obtains artifacts, descriptions, lessons learned, and documentation.
5. STRS repository manager stores the artifacts, descriptions, and documentation with the appropriate permissions using configuration management.
6. STRS repository manager creates an index entry in the table of contents.
7. STRS repository manager stores the version description.
8. STRS repository manager stores the lessons learned.
9. STRS repository manager obtains and stores the STRS compliance test results. To use a common format, a copy of the STRS Compliance Matrix in Appendix F of STRS Compliance Testing, NASA/TM-2011-217266, should be filled in.
10. STRS repository manager stores any licensing, nondisclosure, and distribution agreements.
11. STRS repository manager stores information and/or links to describe any necessary software tools, their licensing, version, options, and purpose.
12. STRS repository manager stores information concerning the operating system for the platform and each tool used including platform, operating system, licensing, version, options, and purpose.

13. STRS repository manager stores the metrics.
    a. Time for implementation
    b. Lines of code and level of effort
    c. NPR 7150.2A classification
    d. Reuse Readiness Level (RRL) and justification
    e. Technology Readiness Level (TRL) and justification
14. STRS repository manager stores:
    a. NPR 7150.2A requirements mapping matrix
    b. NF-1679, Disclosure of Invention and New Technology (Including Software)

8.2 Pull Use Case

1. Project manager and STRS liaison determine that project is ready for make/buy/reuse/port decision.
2. Project manager requests the STRS repository manager whether the sought for functionality exists within the STRS application repository.
3. STRS repository manager queries description for functional match. If match not found, quit this use case.
4. STRS repository manager checks licensing, nondisclosure, and distribution agreements for releasability. If not releasable, quit this use case.
5. STRS repository manager presents the project manager with any restrictions concerning release such as International Traffic in Arms Regulations (ITAR), Export Administration Regulations (EAR), and whether there is a need for nondisclosure agreements (NDA).
6. STRS repository manager retrieves metrics. STRS repository manager presents the project manager the available metrics from which make/buy/reuse cost estimates may be made.
7. Project manager uses metrics and compliance results to decide whether to reuse/port.
8. STRS repository manager marks first project appropriately for reused/ported or not. If not reused or ported, quit this use case.
9. STRS repository manager sets permissions to make the appropriate artifacts available to the STRS application developer for retrieval.
10. STRS repository manager makes first project artifacts and descriptions available to second project to retrieve.
11. The STRS application developer ports the STRS application.
12. The STRS application integrator ports the STRS application.
13. The STRS application developer gives metrics and lessons learned back to the STRS repository manager.
14. Continue with the “push” case in Section 8.1.

9.0 Reuse Readiness Level (RRL)

Reuse readiness level is a metric that can be used to aid in estimating whether software reuse will be efficient and effective. Reuse Readiness Levels are a product of NASA's Earth Science Data Systems Software Reuse Working Group. The following Table 9.1 provides the means of selecting the overall reuse readiness level.

TABLE 9.1.—OVERALL REUSE READINESS LEVEL (RRL)

RRL	Summary	Description
1	Limited reusability; the software is not recommended for reuse.	Little is provided beyond limited source code or precompiled, executable binaries. There is no support, contact information for developers or rights for reuse specified, the software is not extensible, and there is inadequate or no documentation.
2	Initial reusability; software reuse is not practical.	Some source code, documentation, and contact information are provided, but these are still very limited. Initial testing has been done, but reuse rights are still unclear. Reuse would be challenging and cost-prohibitive.
3	Basic reusability; the software might be reusable by skilled users at substantial effort, cost, and risk.	Software has some modularity and standards compliance, some support is provided by developers, and detailed installation instructions are available, but rights are unspecified. An expert may be able to reuse the software, but general users would not.
4	Reuse is possible; the software might be reused by most users with some effort, cost, and risk.	Software and documentation are complete and understandable. Software has been demonstrated in a lab on one or more specific platforms, infrequent patches are available, and intellectual property issues would need to be negotiated. Reuse is possible, but may be difficult.
5	Reuse is practical; the software could be reused by most users with reasonable cost and risk.	Software is moderately portable, modular, extendable, and configurable, has low-fidelity standards compliance, a user manual, and has been tested in a lab. A user community exists, but may be a small community of experts. Developers may be contacted to request limited rights for reuse.
6	Software is reusable; the software can be reused by most users although there may be some cost and risk.	Software has been designed for extensibility, modularity, and portability, but software and documentation may still have limited applicability. Tutorials are available, and the software has been demonstrated in a relevant context. Developers may be contacted to obtain formal statements on restricted rights or to negotiate additional rights.
7	Software is highly reusable; the software can be reused by most users with minimum cost and risk.	Software is highly portable and modular, has high-fidelity standards compliance, provides auto-build installation, and has been tested in a relevant context. Support is developer-organized, and an interface guide is available. Software and documentation are applicable for most systems. Brief statements are available describing limited rights for reuse and developers may be contacted to negotiate additional rights.
8	Demonstrated local reusability; the software has been reused by multiple users.	Software has been shown to be extensible, and has been qualified through test and demonstration. An extension guide and organization-provided support are available. Brief statements are available describing unrestricted rights for reuse and developers may be contacted to obtain formal rights statements.
9	Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems.	Software is fully portable and modular, with all appropriate documentation and standards compliance, encapsulated packaging, a GUI installer, and a large support community that provides patches. Software has been tested and validated through successful use of application output. Multiple statements describing unrestricted rights for reuse and the recommended citation are embedded into the product.

In addition, the reuse readiness may be selected by category. The nine reuse readiness level categories are:

1. Documentation (Table 9.2)
2. Extensibility (Table 9.3)
3. Intellectual Property Issues (Table 9.4)

4. Modularity (Table 9.5)
5. Packaging (Table 9.6)
6. Portability (Table 9.7)
7. Standards Compliance (Table 9.8)
8. Support (Table 9.9)
9. Verification and Testing (Table 9.10)

The tables below contain the RRL level and corresponding description for each category. Choose an RRL level for each category. These may be averaged to estimate an equivalent overall reuse readiness level. For estimating effort and cost for reuse and porting, both overall RRL and detailed RRL by category will be used for estimation until certain ones appear to be more highly correlated with effort and cost actually realized.

TABLE 9.2.—DOCUMENTATION REUSE READINESS LEVEL (RRL)

RRL	Description
1	Little or no internal or external documentation available
2	Partially to fully commented source code available
3	Basic external documentation for sophisticated users available
4	Reference manual available
5	User manual available
6	Tutorials available
7	Interface guide available
8	Extension guide and/or design/developers guide available
9	Documentation on design, customization, testing, use, and reuse is available

TABLE 9.3.—EXTENSIBILITY REUSE READINESS LEVEL (RRL)

RRL	Description
1	No ability to extend or modify program behavior
2	Very difficult to extend the software system, even for application contexts similar to the original application domain
3	Extending the software is difficult, even for application contexts similar to the original application domain
4	Some extensibility is possible through configuration changes and/or moderate software modification
5	Consideration for future extensibility designed into the system for a moderate range of application contexts; extensibility approach defined and at least partially documented
6	Designed to allow extensibility across a moderate to broad range of application contexts, provides many points of extensibility, and a thorough and detailed extensibility plan exists
7	Demonstrated to be extensible by an external development team in a similar context
8	Demonstrated extensibility on an external program, clear approach for modifying and extending features across a broad range of application domains
9	Demonstrated extensibility in multiple scenarios, provides specific documentation and features to build extensions which are used across a range of domains by multiple user groups

TABLE 9.4.—INTELLECTUAL PROPERTY ISSUES REUSE READINESS LEVEL (RRL)

RRL	Description
1	Product developers have been identified, but no rights have been determined.
2	Developers are discussing rights that comply with their organizational policies.
3	Rights agreements have been proposed to developers.
4	Developers have negotiated on rights agreements.
5	Agreement on ownership, limited reuse rights, and recommended citation.
6	Developer list, recommended citation, and rights statements have been drafted.
7	Developer list and limited rights statement included in product prototype.
8	Recommended citation and intellectual property rights statement included in product.
9	Statements describing unrestricted rights, recommended citation, and developers embedded into product.

TABLE 9.5.—MODULARITY REUSE READINESS LEVEL (RRL)

RRL	Description
1	Not designed with modularity
3	Modularity at major system or subsystem level only
5	Partial segregation of generic and specific functionality
7	Clear delineations of specific and reusable components
9	All functions and data encapsulated into objects or accessible through web service interfaces

TABLE 9.6.—PACKAGING REUSE READINESS LEVEL (RRL)

RRL	Description
1	Software or executable available only, no packaging
3	Detailed installation instructions available
5	Software is easily configurable for different context
7	OS detect and auto-build for supported platforms
9	Installation user interface provided

TABLE 9.7.—PORTABILITY REUSE READINESS LEVEL (RRL)

RRL	Description
1	The software is not portable
2	Some parts of the software may be portable
3	The software is only portable with significant costs
4	The software may be portable at a reasonable cost
5	The software is moderately portable
6	The software is portable
7	The software is highly portable
9	The software is completely portable

TABLE 9.8.—STANDARDS COMPLIANCE REUSE READINESS LEVEL (RRL)

RRL	Description
1	No standards compliance
2	No standards compliance beyond best practices
3	Some compliance with local standards and best practices
4	Standards compliance, but incomplete and untested
5	Standards compliance with some testing
6	Verified standards compliance with proprietary standards
7	Verified standards compliance with open standards
8	Verified standards compliance with recognized standards
9	Independently verified standards compliance with recognized standards

TABLE 9.9.—SUPPORT REUSE READINESS LEVEL (RRL)

RRL	Description
1	No support available
2	Minimal support available
3	Some support available
4	Moderate systematic support is available
5	Support provided by an informal user community
6	Formal support available
7	Organized/defined support by developer available
8	Support available by the organization that developed the asset
9	Large user community with well-defined support available

TABLE 9.10.—VERIFICATION AND TESTING REUSE READINESS LEVEL (RRL)

RRL	Description
1	No testing performed
2	Software application formulated and unit testing performed
3	Testing includes testing for error conditions and proof of handling of unknown input
4	Software application demonstrated in a laboratory context
5	Software application tested and validated in a laboratory context
6	Software application demonstrated in a relevant context
7	Software application tested and validated in a relevant context
8	Software application "qualified" through test and demonstration (meets requirements) and successfully delivered
9	Actual software application tested and validated through successful use of application output

10.0 Disclosure of Invention and New Technology NF-1679

For release of software, an NF-1679, Disclosure of Invention and New Technology, or the equivalent, must be submitted. NF-1679 contains the following:

The STRS repository manager must obtain or create NF-1679 Disclosure of Invention and New Technology as well as the Software Release Request Form (SRRA). NF-1679 form requires:

1. Descriptive title
2. Innovator(s) (If multiple innovators, number each to match Box 5.) For each innovator provide:

N	Name	Title	Work Address	Work Phone Number	Work Email Address

3. Innovator's employer when innovation was made (If multiple innovators, number each.) For each innovator provide:

N	Name of Employer	Division and Address of Employer	Organizational Code / Mail Code	Contract / Grant Number

4. Place of performance (Address(es) where innovation made)

N	Street address	Street address (2)	City	State	Zip

5. Employer status (choose one for each innovator) GE = Government, CU = College or University, NP = Non-Profit Organization, SB = Small Business Firm, LE = Large Entity

N	Employer Status

6. Origin (Check all that apply and provide all applicable numbers. If multiple Contracts/Grants, etc., list Contract/Grant Numbers in Box 3 with applicable employer information.)

   [_] NASA In-house including Org. and Mail Code [_____],

   WBS [_____]

   [_] Grant/Cooperative Agreement including Agreement No. [_____], WBS [_____]

   [_] Prime Contract include

   Prime Contract No. [_____], WBS [_____]

   Task No. [_____], and

   Report No. [_____],

   [_] Subcontractor; including Subcontract Tier [_____],

   WBS [_____]

   [_] Joint Effort (contractor, subcontractor and/or grantee contribution(s), and NASA in-house contribution)

   [_] Multiple Effort (multiple contractor, subcontractor and/or grantee contributions, no NASA in-house contribution)

   [_] Other (e.g., Space Act Agreement, MOA) including No. [_____],

   WBS [_____]

7. NASA contracting officer's technical representative (COTR) [_____],

8. Contractor/grantee new technology representative (POC) [_____],

9. Brief abstract (A general description of the innovation which describes its capabilities, but does not reveal details that would enable duplication or imitation of the innovation.)

Section I: Description of the problem or objective that motivated the innovation's development (Enter as appropriate:

A. General description of problem/objective;

B. Key or unique problem characteristics;

C. Prior art, i.e., prior techniques, methods, materials, or devices performing function of the innovation, or previous means for performing function of software;

D. Disadvantages or limitation of prior art.

Section II: Technically complete and easily understandable description of innovation developed to solve the problem or meet the objective (Enter as appropriate; existing reports, if available, may form a part of the disclosure, and reference thereto can be made to complete this description:

A. Purpose and description of innovation/software;

B. Identification of component parts or steps, and explanation of mode of operation of innovation/software preferably referring to drawings, sketches, photographs, graphs, flow charts, and/or parts or ingredient lists illustrating the components;

C. Functional operation;

D. Alternate embodiments of the innovation/software;

E. Supportive theory;

F. Engineering specifications;

G. Peripheral equipment;

H. Maintenance, reliability, safety factors.

Section III: Unique or novel features of the innovation and the results or benefits of its application. Enter as appropriate:

A. Novel or unique features;

B. Advantages of innovation/software;

C. Development or new conceptual problems;

D. Test data and source of error;

E. Analysis of capabilities;

F. For software, any reuse or re-engineering of existing code, use of shareware, or use of code owned by a nonfederal entity.

Section IV: Speculation regarding potential commercial applications and points of contact (Including names of companies producing or using similar products.)

10. Additional documentation (Include copies or list below any pertinent documentation which aids in the understanding or application of the innovation (e.g., articles, contractor reports, engineering specs, assembly/manufacturing drawings, parts or ingredients list, operating manuals, test data, assembly/manufacturing procedures, etc.).)
11. Degree of technology significance (Which best expresses the degree of technological significance of this innovation? Modification to Existing Technology, Substantial Advancement in the Art, or Major Breakthrough)
12. State of development
13. Patent status (Prior patent on/or related to this innovation. (Application Filed, Application No., Application Date), (Patent Issued, Patent No., Issue Date))
14. Indicate the date or the approximate time period which this innovation was developed (i.e., conceived, constructed, tested, etc.)
15. Previous or contemplated publication or public disclosure including dates
    Provide as applicable:

    A. Type of publication or disclosure, e.g., report, conference or seminar, oral presentation;

    B. Disclosure by NASA or Contractor/Grantee;

    C. Title, volume no., page no., and date of publication.

16. Questions for software only

(a) Using non-NASA employees to beta-test the program?
Yes No
If Yes, done under a beta-test agreement?
Yes No

(b) Modification of this program continued by civil servant and/or contractual agreement?
Yes No

(c) Copyright registered?
Yes No
If Yes, then by whom?

(d) Has the latest version been distributed outside of NASA or contractor?
Yes No
If Yes, date of first disclosure:

(e) Were prior versions distributed outside of NASA or Contractor?
Yes No
If Yes, supply NASA or contractor contract:

(f) Contains or based on code not owned by U.S. Government or its contractors?
Yes No
If Yes, name of code and code's owner.
Has a license for use been obtained?
Yes No

17. Development history (stage of development, date, location, identify supporting witnesses (NASA in-house only)

Stage of Development	*Date*	*Location*	*Supporting Witnesses*
First disclosure to others			
First sketch, drawing, logic chart or code			
First written description			
Completion of first model of full size device (invention) or beta version (software)			
First successful operational test (invention) or alpha version (software)			

a. Contribution of innovators (if jointly developed, provide the contribution of each innovator) :

N	Innovator	Percent Contribution

b. Indicate any past, present, or contemplated government use of the innovation

18. Signatures of innovators, witnesses, and NASA Approvals

N	Innovator	Signature	Date
N	Witness		Date
1			
2			

11.0 NASA Software Release Request Authorization

The NASA Software Release Request Requisition Authorization (SRRA) must accompany the NF-1679. Since the SRRA has not been published, a list of the requested data from a 7/26/2011 version is included to inform the reader of the information required.

1. Date of Request:
2. Full Name of Requestor:
3. Software Title and Abbreviation:
4. Technology Case Number:
5. Version Number:
6. Version Date:
7. Technical Point of Contact (Person Who Knows the Most About the Software):
    a. Full Name:
    b. Company Name:
    c. Company Address:
    d. Mail Code:
    e. Organization Code:
    f. Phone:
    g. E-Mail Address:
8. Government Point of Contact (If Technical Point of Contact Is a Contractor):
    a. Full Name:
    b. Agency Name:
    c. Mailing Address:
    d. Organization Code:
    e. Phone:
    f. E-Mail Address:
9. Brief Description of Software:
10. What Type of Code Will Be Released? Executable, Source, Executable and Source
11. Type of Release Requested:
    a. Government Purpose Only
    b. U.S. Release (Recipient must be U.S. Person or Company)
    c. U.S. and Foreign Release (All U.S. persons and allowed foreign nationals)
    d. Public Release
    e. Open Source Release (No Release Restrictions)

12. Will A User Manual Be Released With Your Software?
    Yes No
13. How Do You Plan to Distribute Your Software? (i.e., CD-ROM, E-Mail Attachment, Download After E-Mailing Password)
14. Are There Any Programmatic Restrictions On Release of Your Software?
    Yes No
15. If Yes, Explain:
16. What Is The Classification And Safety Critical Designation Of The Software?

    NOTE: Refer to NPR 7150.2A, Appendix E and NASA-STD-8739.8, Appendix A for an explanation of the classifications and safety critical designations for software.
17. Does the Software Comply With the Software Engineering and Assurance Requirements of NPR 7150.2A, and NASA-STD-8739.8, Software Assurance Standard, for the Applicable Software Classification? Yes No

    IMPORTANT: Please use the "Instructions for Use of Compliance Matrices for Software Classifications" file to complete the appropriate matrix for the class of software to be released. Attach a copy of the completed matrix to this document when submitting it for review. Questions concerning applicability of requirements should be directed to the local designated Software Engineering Technical Authority (for NPR 7150.2A) or Software Assurance Technical Authority (for NASA-STD-8739.8).
18. If Software Does NOT Comply, Are the Deviations/Waivers Documented and Approved?
    Yes No
    (Please Attach Relevant Deviations/Waivers)
19. Is the Software Safety-Critical as Defined In NASA-STD-8739.8?
    Yes No
20. If Yes, Does It Comply With the Software Safety Requirements of NASA-STD-8719.13, Software Safety Standard?
    Yes No
21. If No, Are the Deviations/Waivers Documented and Approved?
    Yes No
    (Please Attach Relevant Deviations/Waivers)
22. What Is the Software's Technology Readiness Level (TRL) as Defined in NPR 7120.8, NASA Research and Technology Program and Project Management Requirements? TRL Level:
    (See Attachment 2 - TRL Level Chart or see TRL calculator)
23. Is the Software Section 508 Compliant?
    Yes No
24. Does Your Software Include Any Embedded Computer Databases?
    Yes No
25. If Yes, Explain:
26. Does Your Software Use or Call Any Software or Libraries?
27. Open Source:
    Yes No

28. Proprietary/Commercial:
    Yes No
29. If Yes, List the Items Used, Under What License They Were Obtained, and the URL for the License:
30. Are There Any Known Export Restrictions That Apply to the Software?
    Yes No
31. If Yes, Explain (e.g., EAR or ITAR Controlled):
32. Was Software Development Funded By the Military?
    Yes No
33. If Yes, Explain Predominant Application(s) ? (Military, Civil, or Both):
34. Does Your Software Contain Embedded Firewall Information or Require Ports to be Opened in the Firewall for Proper Operation?
    Yes No
35. If Yes, Explain:
36. Does Your Software Contain Embedded Credentials (e.g., Username/Password, Certificates, Encryption Keys)?
    Yes No
37. If Yes, Explain:
38. Does Your Software Analyze Network Traffic?
    Yes No
39. If Yes, Explain:
40. Does Your Software Use or Include Encryption?
    Yes No
41. If Yes, Explain:
42. Has the Software Application Data Owner Been Consulted to Ensure that Your Software Documentation, Embedded Files, Code, or Other Artifacts Do Not Contain Residual SBU Data?
    Yes No
43. If No, Explain:
44. Has the Applicable Center Privacy Manager Been Consulted to Determine if Your Software Documentation, Embedded Files, Code, or Other Artifacts Contain Any Personally Identifiable Information (PII)?
    Yes No
45. A Frequently Asked Questions (FAQ) Document Addressing NASA PII Can Be Found at: http://insidenasa.nasa.gov/ocio/information/info_privacy/pii_faq.html
46. If No, Explain:
47. If Your Software is Safety-Critical as Defined in NASA-STD-8739.8 or Release of Type General Public or Open Source as Defined in this Form, has a Code Review Been Performed to Discover Any Residual Security and Privacy Risks?
    Yes No

48. If You Do Not Possess the Resources to Perform this Review, Please Contact Your Center ITSM or the Agency OCIO (hq-dl-itspm@mailnasa.gov) for Assistance.

49. If No, Explain:

The technical point of contact, as listed on the first page of the SRRA form, reviews the SRRA form for technical concurrence and recommendations. The NASA POC is the NASA employee most familiar with the software (could be the COTR for a NASA contract/grant).

The NASA lead for the project/program under which the software was developed reviews the SRRA form for concurrence and recommendations. If the software isn't specific to a project or program, this person would be the NASA manager for the organization responsible for creation of the software.

12.0 Technology Readiness Level (TRL)

Technology readiness level (TRL) is a metric that can be used to assess the maturity of evolving technologies. The technology readiness level may be selected using the following Table 12.1:

TABLE 12.1.—TECHNOLOGY READINESS LEVEL (TRL)

TRL	Description
1	**Basic principles observed and reported:** Transition from scientific research to applied research. Essential characteristics and behaviors of systems and architectures. Descriptive tools are mathematical formulations or algorithms.
2	**Technology concept and/or application formulated:** Applied research. Theory and scientific principles are focused on specific application area to define the concept. Characteristics of the application are described. Analytical tools are developed for simulation or analysis of the application.
3	**Analytical and experimental critical function and/or characteristic proof-of-concept:** Proof of concept validation. Active Research and Development (R&D) is initiated with analytical and laboratory studies. Demonstration of technical feasibility using breadboard or brassboard implementations that are exercised with representative data.
4	**Component/subsystem validation in laboratory environment:** Standalone prototyping implementation and test. Integration of technology elements. Experiments with full-scale problems or data sets.
5	**System/subsystem/component validation in relevant environment:** Thorough testing of prototyping in representative environment. Basic technology elements integrated with reasonably realistic supporting elements. Prototyping implementations conform to target environment and interfaces.
6	**System/subsystem model or prototyping demonstration in a relevant end-to-end environment (ground or space):** Prototyping implementations on full-scale realistic problems. Partially integrated with existing systems. Limited documentation available. Engineering feasibility fully demonstrated in actual system application.
7	**System prototyping demonstration in an operational environment (ground or space):** System prototyping demonstration in operational environment. System is at or near scale of the operational system, with most functions available for demonstration and test. Well integrated with collateral and ancillary systems. Limited documentation available.
8	**Actual system completed and "mission qualified" through test and demonstration in an operational environment (ground or space):** End of system development. Fully integrated with operational hardware and software systems. Most user documentation, training documentation, and maintenance documentation completed. All functionality tested in simulated and operational scenarios. Verification and Validation (V&V) completed.
9	**Actual system "mission proven" through successful mission operations (ground or space):** Fully integrated with operational hardware/software systems. Actual system has been thoroughly demonstrated and tested in its operational environment. All documentation completed. Successful operational experience. Sustaining engineering support in place.