Reuse Readiness Levels (RRLs)

Prepared by:
NASA Earth Science Data Systems –
Software Reuse Working Group

April 30, 2010

Version 1.0

Earth Science Data Systems Software Reuse Working Group

Editor:

James J. Marshall (INNOVIM / NASA Goddard Space Flight Center)

Contributing Working Group Members:

Stephen Berrick (NASA Goddard Space Flight Center)

Angelo Bertolli (INNOVIM / NASA Goddard Space Flight Center)

Corey Bettenhausen (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center)

Howard Burrows (Autonomous Undersea Systems Institute / National Science Digital Library)

Saurabh Channan (University of Maryland)

Victor Delnore * (NASA Langley Research Center)

Robert R. Downs ** (Columbia University / NASA Socioeconomic Data and Applications Center)

Yonsook Enloe (SGT Inc. / NASA Goddard Space Flight Center)

Stefan Falke (Washington University in St. Louis)

Al Fleig * (PITA Analytic Sciences / NASA Goddard Space Flight Center)

Michael Folk (National Center for Supercomputing Applications)

Neil Gerard (INNOVIM / NASA Goddard Space Flight Center)

Ryan Gerard (INNOVIM / NASA Goddard Space Flight Center)

Larry Gilliam (INNOVIM / NASA Goddard Space Flight Center)

Mary Hunter (INNOVIM / NASA Goddard Space Flight Center)

Tommy Jasmin (University of Wisconsin-Madison, Space Science and Engineering Center)

Louis Kouvaris (Science Applications International Corporation / NASA Goddard Space Flight Center)

Michael Leyton (Rutgers University)

Chris A. Mattmann ** (NASA Jet Propulsion Laboratory)

David McComas (NASA Goddard Space Flight Center)

Neal Most (INNOVIM / NASA Goddard Space Flight Center)

Shahin Samadi (INNOVIM / NASA Goddard Space Flight Center)

Mark Sherman (SGT Inc. / NASA Goddard Space Flight Center)

Ross Swick (National Snow and Ice Data Center, University of Colorado – Boulder)

Curt Tilmes (NASA Goddard Space Flight Center)

Petr Votava (SGE / NASA Ames Research Center)

Christine Whalen (INNOVIM / NASA Goddard Space Flight Center)

Bruce Wilson (Oak Ridge National Laboratory)

Robert Wolfe * (NASA Goddard Space Flight Center)

* Past Co-chairs

** Present Co-chairs

Working Group Participants:

Nadine Alameh (MobiLaps LLC)

Bradford Castalia (University of Arizona)

Bill Frakes (Virginia Tech)

Emily Greene (Raytheon Company)

Gary Jackson (University of Maryland)

Virginia Kalb (NASA Goddard Space Flight Center)

Kwo-Sen Kuo (University of Maryland Baltimore County / NASA Goddard Space Flight Center)

Steve Olding (Everware / NASA Goddard Space Flight Center)

Margaret Pippin (NASA Langley Research Center)

Bill Teng (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center)

Fred Watson (California State University, Monterey Bay)

Jonathan Wilmot (NASA Goddard Space Flight Center)

Note: All affiliations above were recorded at the time of the member's contribution or participation.

Acknowledgement:

The ESDS Software Reuse Working Group would like to acknowledge the work of all past members of the Working Group who contributed to the development of the Reuse Readiness Levels presented in this document.

Recommended Citation:

NASA Earth Science Data Systems Software Reuse Working Group (2010). Reuse Readiness Levels (RRLs), Version 1.0. April 30, 2010. Available: http://www.esdswg.org/softwarereuse/Resources/rrls/

Table of Contents

1. Introduction

Objective

Recognizing the need to measure the maturity of software for reuse, the NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG) proposes a set of Reuse Readiness Levels (RRLs). The maturity of a particular technology can be measured in various ways, one common method being with Technology Readiness Levels (TRLs) or other similar measurements. However, the ability or readiness of a particular technology to be reused is generally not considered, or plays only a small role if it is considered.

Context

The ESDS Working Groups (WGs) were established by NASA in 2004 as a follow-on to the Strategic Evolution of Earth Science Enterprise Data Systems (SEEDS) Study, which built on New Data and Information Systems and Services (NewDISS) concepts of distributed, heterogeneous, measurement-based data systems. Together, these studies emphasize the concept of flexible, distributed system elements, which have considerable implementation freedom, the role of the community in shaping data system practices, and a focus on competition, measurement-based (Principal Investigator) systems, and an evolutionary approach to overall data system management. Four WGs were convened to carry out this charter: (1) Metrics Planning and Reporting, (2) Reuse and Reuse Frameworks, (3) Standards and Interfaces, and (4) Technology Infusion. These groups provide community forums for bringing issues to the table in these topic areas, and separately and jointly identify methods, tools, and resources that offer improvements and solutions to recognized issues. Although the WGs do not make policy, they do send recommendations to NASA Headquarters for consideration.

Background

In 1995, John C. Mankins wrote a white paper on Technology Readiness Levels (TRLs) [1], an approach to assessing the maturity of technology that also allows for the consistent comparison of maturity between different types of technology. His paper was intended to make the general model as useful as possible by having the maturity measurement span basic research in new technologies to system launch and operations. He describes a set of nine levels, each with a short summary label and a longer clarifying description. While they tend to focus on the hardware aspect of technology, these TRLs, adopted by NASA, have also been applied to software [2], and recent work has revised them to include descriptions for both hardware and software along with exit criteria for moving from one level to the next (NPR 7120.8, Appendix J). These are one of the commonly used definitions of TRLs.

The Department of Defense (DoD) also developed a set of TRLs, and their Technology Readiness Assessment (TRA) Deskbook [3], released in May 2005 and updated in 2009, includes a set of definitions, descriptions, and supporting information about their Software TRLs. The DoD also has TRLs related to hardware and manufacturing. The DoD

definitions of TRLs are similar to those of NASA, and are another commonly used definition for the TRL scale. A number of other organizations have developed similar scales for assessing technology maturity [4–10], and links to them are provided in the References section at the end of this document.

Justification

In its review of potential measures for software reuse readiness, the NASA Earth Science Data System (ESDS) Software Reuse Working Group (referred to as WG throughout the document) has examined the measures of technology and software maturity mentioned above and in the References section, which were selected because they generally focus on measuring the maturity of software, and found that these typically do not consider reuse/reusability in their measures. The emphasis of such measures appears to focus on the maturity of the technology or software as a whole. When reusability of software and related artifacts is considered, it is frequently in a limited manner. For example, the Open Process Framework's Technology Readiness Assessment [5] does include reuse, but only in terms of reused critical technologies. The reuse of non-critical technologies is not addressed.

The WG has identified the lack of attention paid to the reusability of technology as a shortcoming of existing TRLs and other measurements of technology maturity. When the WG surveyed members of the Earth science community on their reuse practices and experiences, the results showed that the reuse of existing technologies is most often performed in order to save time, save money, and ensure the reliability of the product [11]. Having a framework tool to assess and quantify a software asset's readiness to be reused would be a valuable aid to reusers in achieving these goals. It also could help software adopters to understand what the asset will offer and provide a sense of how much modification may be necessary to meet their needs. If a measure of reusability were included as an element of metadata for assets in software catalogs and repositories, it could assist potential reusers in making decisions on which software assets to reuse. While this would not eliminate the need for the complete testing of candidate software assets to determine their viability as a solution, it could assist software developers in narrowing down the number of potential solutions to be fully examined and tested. The availability of such measures could increase the efficiency of the process by which developers find, assess, and select reusable software assets for reuse in their projects [12].

Therefore, the WG recommends the adoption of Reuse Readiness Levels (RRLs) that specifically address the maturity of software in the sense of reusability as a means for encouraging and enabling software reuse, within the Earth science community and within other communities that use software to complete their objectives.

2. Development of the RRLs

Identification of Topic Areas

Recognizing that the existing TRLs are a useful measure and have been successfully applied within their domain, the WG decided to use NASA's TRLs [1] as a guide while

developing RRLs. In particular, the WG agreed to have RRLs ranging from 1 to 9 (inclusive), to align with the familiar TRL scale. Through extensive discussions, the WG identified the nine topic areas that were deemed important for measuring the reuse maturity of software. Alphabetically, they are:

- Documentation
- Extensibility
- Intellectual Property Issues
- Modularity
- Packaging
- Portability
- Standards Compliance
- Support
- Verification and Testing

These topic areas are defined and described in more detail in Section 5.

Development of Topic Area Levels and Reuse Readiness Levels

Volunteers from the WG drafted an initial set of levels for each topic area, with at least two people contributing to each topic. In most areas, volunteers were able to identify nine levels of reuse maturity, but in some areas, fewer levels were identified. These levels were spread out roughly evenly on the 1 to 9 scale used by the RRLs. Once the topic level drafts were completed, members of the WG began drafting potential RRLs by summarizing each level from 1 through 9 across all nine topic areas. At the ESDS Working Group Meeting in 2007, the WG discussed the topic levels and their summaries, and developed a draft of RRL summaries. These shorter labels and the topic area levels were presented to members of all four ESDS Working Groups during a plenary session [13] and the full set of topic area levels were presented during a poster session [14]. Feedback was received on the draft at this state at both presentations. Many valuable comments were received, and all involved in the discussions agreed that the RRLs would be a valuable tool once they were more fully developed.

The WG continued to work on the development of the RRLs during the end of 2007, and drafted a set of longer descriptions to accompany the shorter labels that were drafted at the ESDS Working Group Meeting. This work was presented at the 2007 AGU Fall Meeting [15], where additional feedback was received. Also, a discussion held during the 2008 Winter ESIP Federation Meeting [16] provided further comments about the RRLs for the WG to consider.

To address the feedback, comments, and suggestions received, the WG reviewed the topic area levels in early 2008 with those remarks in mind. Volunteers reviewed topic area levels that they did not help write, and the suggestions for improvements were sent to the original authors of each topic level to address. Some feedback applied generally to multiple topic areas; these were also addressed by the original topic level authors during their revisions. Updates took the form of revisions to the topic area levels themselves and/or the addition of explanatory text. After the topic area levels were revised, the WG again looked at the RRLs created as a summary of the topic area levels, and updated them to remain aligned with the topic area levels. During multiple breakout sessions of the 2008 ESDS Working Group Meeting, another revision of the RRL topic area levels was

performed. A workshop of the ESDS Working Group was convened in 2010 to finalize the RRL document, resulting in the Reuse Readiness Levels (RRLs), Version 1.0.

3. Summary of RRLs

Following the format of TRLs, there are nine Reuse Readiness Levels (RRLs) ranging from 1 (least mature) to 9 (most mature).

- RRL 1 – *Limited reusability; the software is not recommended for reuse.*
- RRL 2 – *Initial reusability; software reuse is not practical.*
- RRL 3 – *Basic reusability; the software might be reusable by skilled users at substantial effort, cost, and risk.*
- RRL 4 – *Reuse is possible; the software might be reused by most users with some effort, cost, and risk.*
- RRL 5 – *Reuse is practical; the software could be reused by most users with reasonable cost and risk.*
- RRL 6 – *Software is reusable; the software can be reused by most users although there may be some cost and risk.*
- RRL 7 – *Software is highly reusable; the software can be reused by most users with minimum cost and risk.*
- RRL 8 – *Demonstrated local reusability; the software has been reused by multiple users.*
- RRL 9 – *Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems.*

4. Descriptions of RRLs

The longer descriptions of each RRL are provided below, along with the shorter summary presented in the previous section. These descriptions are intended to clarify the meaning of the summaries and provide more details, helping to ensure that the assessments of software reusability are performed in a consistent manner.

RRL 1

Limited reusability; the software is not recommended for reuse.
Little is provided beyond limited source code or pre-compiled, 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.

RRL 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.

RRL 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.

RRL 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.

RRL 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.

RRL 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.

RRL 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.

RRL 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.

RRL 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.

5. Definitions of Topic Areas

The previously described RRLs were created by combining levels for individual topic areas that the WG deemed important for measuring reuse maturity. These nine areas were documentation, extensibility, intellectual property issues, modularity, packaging, portability, standards compliance, support, and verification/testing, and they are defined below.

Documentation

Information that describes the software asset and how to use it.

Documentation consists of installation and developer guides, development methodologies and documentation of the support available, API specifications, commented code and build instructions, technical support instructions and support forums, technical manuals, libraries and tutorials, and reuse and deployment case studies. This documentation may be in various stages of development and accessibility, and may not have a clear audience defined.

Documentation enables potential adopters to determine whether the software addresses the need and informs adopters how to utilize the software and reduce the risks and costs of reuse. Documentation includes descriptions of interfaces and capabilities, information about the execution environment, and instructions for the consumer on the purpose of the asset and on ways in can be reused. Documentation also describes plans for subsequent releases and future development.

Extensibility

The ability of the asset to be grown beyond its current context.

The implementation takes into consideration future growth and ease of extending function. A measure of the ability to extend a system and the level of effort required to implement the extension. Extensions, or expandability, can apply to re-engineering or during runtime.

Extensibility is an important dimension to be able to incorporate an asset and add to or modify its functionality.

Intellectual Property Issues

The legal rights for obtaining, using, modifying and distributing the asset.

A formal and documented explanation of the involved parties and roles, with binding statements describing any licensing mechanisms, ownership rights, restrictions, and user/consumer responsibilities related to the distribution and reuse of assets. The legal rights are established in accordance with the policies and laws of the organization that originally produced the software.

Potential adopters need to understand the intellectual property issues to know whether they have the authority to reuse the software.

Modularity

The degree of segregation and containment of an asset or components of an asset.

Modularity is a software design technique that increases the extent to which software is composed from separate components, called modules. Conceptually, modules represent a separation of and encapsulation of concern, purpose, and function, and they improve maintainability and reusability.

Modular assets generally are easier to synthesize and extend.

Packaging

The methodology and technology for assembling and encapsulating the components of a software asset.

Packaging pertains to the technologies, standards, and procedures related to gathering, organizing, assembling, and compressing the parts of a software system and distributing it as a collection.

Packaging is important to ensure completeness, to allow distribution, and to simplify the installation of the asset.

Portability

The independence of an asset from platform-specific technologies.

Portability refers to two components: software consisting of source code that can be compiled for various computing platforms; software executables that can be executed on various platforms.

The ability to be installed or executed on various platforms maximizes reuse potential and increases the flexibility and (re-)usability of the asset and its applications.

Standards Compliance

The adherence of an asset to accepted technology definitions.

Concerning commonly accepted criteria, models, patterns and/or specifications have been followed in the creation of a reusable asset; and at what level the asset complies with the standard.

By complying with accepted standards, the asset has increased potential for adoption.

Support

The amount and type of assistance available to users of the asset.

Technical support exists, in the form of various communication methods with the asset's developers, documentation/knowledge bases, user communities, support level agreements, and online forums. A release strategy and plan for patches and versions has been created.

Support provisions expertise to assist in maintenance, evolution, extension and issue resolution.

Verification and Testing

The degree to which the functionality and applicability of the asset has been demonstrated.

This can be realized through the provision of test material, requirements compliance, proper function, and usability (robustness). Tests documented, results analyzed and published, and fixes and enhancements applied.

Sufficient verification and testing increases the accuracy and confidence and reduces potential risks and costs of reuse.

6. Summary of Topic Areas Levels

A summary of all topic area levels is provided in Table 1. A detailed description of each topic area level follows in Section 7.

Table 1 – Reuse Readiness Level (RRL) Topic Area Levels Summary

	Documentation	Extensibility	Intellectual Property Issues	Modularity	Packaging	Portability	Standards Compliance	Support	Verification and Testing
Level 1	Little or no internal or external documentation available	No ability to extend or modify program behavior	Product developers have been identified, but no rights have been determined.	Not designed with modularity	Software or executable available only, no packaging	The software is not portable	No standards compliance	No support available	No testing performed
Level 2	Partially to fully commented source code available	Very difficult to extend the software system, even for application contexts similar to the original application domain	Developers are discussing rights that comply with their organizational policies.			Some parts of the software may be portable	No standards compliance beyond best practices	Minimal support available	Software application formulated and unit testing performed
Level 3	Basic external documentation for sophisticated users available	Extending the software is difficult, even for application contexts similar to the original application domain	Rights agreements have been proposed to developers.	Modularity at major system or subsystem level only	Detailed installation instructions available	The software is only portable with significant costs	Some compliance with local standards and best practices	Some support available	Testing includes testing for error conditions and proof of handling of unknown input
Level 4	Reference manual available	Some extensibility is possible through configuration changes and/or moderate software modification	Developers have negotiated on rights agreements.			The software may be portable at a reasonable cost	Standards compliance, but incomplete and untested	Moderate systematic support is available	Software application demonstrated in a laboratory context
Level 5	User manual available	Consideration for future extensibility designed into the system for a moderate range of application contexts; extensibility approach defined and at least partially documented	Agreement on ownership, limited reuse rights, and recommended citation.	Partial segregation of generic and specific functionality	Software is easily configurable for different contexts	The software is moderately portable	Standards compliance with some testing	Support provided by an informal user community	Software application tested and validated in a laboratory context
Level 6	Tutorials available	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	Developer list, recommended citation, and rights statements have been drafted.			The software is portable	Verified standards compliance with proprietary standards	Formal support available	Software application demonstrated in a relevant context
Level 7	Interface guide available	Demonstrated to be extensible by an external development team in a similar context	Developer list and limited rights statement included in product prototype.	Clear delineations of specific and reusable components	OS detect and auto-build for supported platforms	The software is highly portable	Verified standards compliance with open standards	Organized/defined support by developer available	Software application tested and validated in a relevant context
Level 8	Extension guide and/or design/developers guide available	Demonstrated extensibility on an external program, clear approach for modifying and extending features across a broad range of application domains	Recommended citation and intellectual property rights statement included in product.				Verified standards compliance with recognized standards	Support available by the organization that developed the asset	Software application "qualified" through test and demonstration (meets requirements) and successfully delivered
Level 9	Documentation on design, customization, testing, use, and reuse is available	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	Statements describing unrestricted rights, recommended citation, and developers embedded into product.	All functions and data encapsulated into objects or accessible through web service interfaces	Installation user interface provided	The software is completely portable	Independently verified standards compliance with recognized standards	Large user community with well-defined support available	Actual software application tested and validated through successful use of application output

7. Description of Topic Areas Levels

The detailed topic area levels described here are labeled as "Level", but their numbering corresponds to the overall RRLs. This resulted in some topic areas with less than nine levels missing some level numbers, as the levels present were spread out over the entire nine-level range of the overall RRLs.

Documentation

Level 1 – Little or no internal or external documentation available.

Source code is available, with little or no useful internal or external documentation.

Level 2 – Partially to fully commented source code available.

Source code is available and fully commented, but no other documentation is provided. It may be challenging for a good programmer to determine how to reuse the software.

Level 3 – Basic external documentation for sophisticated users available.

For example, a README file, a "man" page, or command line usage examples. This type of documentation would be sufficient for a sophisticated user to figure out how to use the software, but probably not a general user.

Level 4 – Reference manual available.

Reference manual provides complete documentation on use of the software, but may not be easily approached or accessed by general users. Some documentation relevant to customization is available.

Level 5 – User manual available.

User manual allows a "normal" or general user to understand how to use and possibly customize aspects of the software.

Level 6 – Tutorials available.

Step-by-step walkthroughs of how the software is customized and used in various scenarios, demos, etc. This makes it very easy to understand/teach the software and use it in a new project.

Level 7 – Interface guide available.

Documentation describes how to customize and interface the software with other software, programmatic interfaces, APIs, etc., so that it can more easily be embedded in a larger system.

Level 8 – Extension guide and/or design/developers guide available.

An extension guide provides information on how to customize and add to the software, add plug-ins and the like, use internal programming "languages", etc. A design/developers guide provides a description of internals, design documentation, internal documentation, etc. that is sufficient for someone "skilled in the art" to contribute to the development of the software or take over maintenance of the software.

Level 9 – Documentation on design, customization, testing, use, and reuse is available.

All stages of the software engineering lifecycle are fully documented. This includes design and review artifacts, testing artifacts, customization, and regression tests. The documentation provided is easy to read/access and is appropriate for different categories of users.

Extensibility

Level 1 – No ability to extend or modify program behavior.

Source code is not available; execution parameters cannot be changed, and/or it is not possible to extend the functionality of the software, even for application contexts similar to the original application domain.

Level 2 – Very difficult to extend the software system, even for application contexts similar to the original application domain.

The software was not designed with extensibility in mind. While some level of documentation and/or source code is available, it is extremely difficult to extend the software. For cases where source code is available, the logical flow of code may be hard to follow, with few (if any) comments, and little to no cohesion.

Level 3 – Extending the software is difficult, even for application contexts similar to the original application domain.

Minimal consideration to extensibility is included in the design, through use of methods such as object-oriented design or other tools which provide logical cohesion. Where source code is available, the software has some structure, but may have a high number of independent logical paths, minimal comments and documentation, and/or a low degree of cohesion.

Level 4 – Some extensibility is possible through configuration changes and/or moderate software modification.

Consideration to extensibility to some range of application contexts is included in the design though means such as (a) use of configuration files, (b) isolation of configuration parameters and constants in clearly identified sections of source code (distinct from logic and display code), (c) some documentation of the effects of changes to these parameters and the allowed values for these parameters, and/or (d) effective use of programming practices designed to enable reuse, such as object oriented design.

Level 5 – Consideration for future extensibility designed into the system for a moderate range of application contexts; extensibility approach defined and at least partially documented.

The procedures for extending the software are defined, whether by source code modification (e.g., object-oriented design) or through the provision of some type of extension functionality (e.g., callback hooks or scripting capabilities). Where source code modification is part of the extension plan, the software is well-structured, has a moderate to high level of cohesion, and has configuration elements clearly separated from logic and display elements. Internal and external documentation are sufficient to allow an experienced programmer to understand program flow and logic with moderate effort.

Level 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.

The extensibility capability for the software is well defined, sufficient to enable an experienced developer generally familiar with the project to extend the software. That documentation should include clear information about the range of application contexts to which the software can be extended as well as potential limitations on expansion.

Level 7 – Demonstrated to be extensible by an external development team in a similar context.

The software has been extended and applied to a similar application context to the original. This extension may have been done by an external team using extension documentation, by may have involved substantial assistance from the original development team members.

Level 8 – Demonstrated extensibility on an external program, clear approach for modifying and extending features across a broad range of application domains.

The software has been extended by at least one group of users outside the original development group using existing documentation and with no assistance from the original development team.

Level 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.

The software is regularly extended externally by users across multiple applications using available documentation. There may be a library available of user-generated content for extensions.

Intellectual Property Issues

Level 1 – Developers have been identified, but no rights have been determined.

Product developers have been identified and their responsibilities have been determined, but they have not considered or determined the rights for the product.

Level 2 – Developers are discussing rights that comply with their organizational policies.

Relevant policies of developers have been reviewed for applicability to intellectual property rights, but no agreements have been proposed. Rights are not specified.

Level 3 – Rights agreements have been proposed to developers.

Each developer has received a draft intellectual property rights agreement that would result from cooperative activities with other developers. Rights are not specified.

Level 4 – Developers have negotiated on rights agreements.

Developers have reviewed proposals from each of the other developers and have proposed an agreement that addresses any potential conflicts in the proposed intellectual property rights and responsibilities for development. A limited rights statement has been drafted and developers may be contacted to negotiate rights for reuse.

Level 5 – Agreement on ownership, limited reuse rights, and recommended citation.

Developers have agreed on proposed ownership, limited intellectual property rights for reuse, and responsibilities. Order of developers' names, recommended citation, and agreements have been formalized. Developers may be contacted to obtain formal statements on restricted rights for reuse.

Level 6 – Developer list, recommended citation, and rights statements have been drafted.

Agreements on development responsibilities, the list of developers, a recommended citation, and intellectual property rights statements, offering limited rights for reuse have been drafted and are included in package. Developers may be contacted to obtain formal statements on restricted rights or to negotiate additional rights.

Level 7 – Developer list and limited rights statement included in product prototype.

A list of developers, recommended citation, and intellectual property rights statements, including copyright or ownership statements, are embedded in the source code of the product, in the documentation, and in the expression of the software upon execution. These include any legal language that has been approved by all parties or their representatives, machine-readable code expressing intellectual property, and concise statements in language that can be understood by laypersons, such as a pre-written, recognizable license. Brief statements are available describing limited rights, restrictions, and conditions for reuse. Developers may be contacted to negotiate additional rights.

Level 8 – Recommended citation and intellectual property rights statement included in product.

All parties have reviewed the list of developers, recommended citation, and intellectual property rights statements, including limited rights for reuse, in the product to ensure that all interests are represented and that the statements conform to their institutional policies and agreements. Brief statements are available describing unrestricted rights and any conditions for reuse. Developers may be contacted to obtain formal rights statements.

Level 9 – Statements describing unrestricted rights, recommended citation, and developers embedded into product.

Multiple statements are embedded into the product describing unrestricted rights and any conditions for reuse, including commercial reuse, and the recommended citation. The list of developers is embedded in the source code of the product, in the documentation, and in the expression of the software upon execution. The intellectual property rights statements are expressed in legal language, machine-readable code, and in concise statements in language that can be understood by laypersons, such as a pre-written, recognizable license.

Modularity

Level 1 – Not designed with modularity.

Research or prototype-grade code written with no designs for organizing code in terms of functionality for modularity or reuse.

Level 3 – Modularity at major system or subsystem level only.

No clear distinctions between generic and solution-specific functionality; few internal functions accessible by external programs (i.e., closed architecture), limited distinction between visible functions; code is organized into a primary system that provides general functionality and one or two subsystems that each provide multiple, unrelated, functions; code within each module contains many independent logical paths.

Level 5 – Partial segregation of generic and specific functionality.

Top to bottom structuring into individual components that provide functions or services to outside entities (i.e., open architecture); internal functions or services documented, but not consistently; modules have been created for generic functions, but modules have not been created for all of the specified functions; code within each module contains many independent logical paths.

Level 7 – Clear delineations of specific and reusable components.

Organization of all components into libraries or service registries; consistent documentation of all libraries as APIs or standard web service interfaces; modules have been created for all specified functions and organized into libraries with consistent features within interfaces; code within each module contains many independent logical paths.

Level 9 – All functions and data encapsulated into objects or accessible through web service interfaces.

All functions and data encapsulated into objects or accessible through web service interfaces; consistent error handling; use of generic extensions to program languages for stronger type checking and compilation-time error checking; services available externally, e.g., in “third-party” service workflows; code within each module contains few independent logical paths.

Packaging

Level 1 – Software or executable available only, no packaging.

Inadequate or no documentation and no auto-build/install facility is available.

Level 3 – Detailed installation instructions available.

System includes auto-build feature, but is built for a particular operating system.

Level 5 – Software is easily configurable for different contexts.

For example, locations of resources (files, directories, URLs) are configurable. All configuration-specific information is centralized.

Level 7 – OS-detect and auto-build for supported platforms available.

Operating system detection configuration files are available. Packaging includes auto-build for supported OS platforms and suite of regression tests for platform-specific testing.

Level 9 – Installation user interface provided.

A user interface guides the installer through all steps needed to build, configure, and install the software.

Portability

Level 1 – The software is not portable.

No source code or instructions for customization are provided. Executable binaries are provided and there are known severe limitations for running it on the hardware or operating system. There is only minimal information on installation or use. There is no information on porting to another platform or application.

Level 2 – Some parts of the software may be portable.

Some source code is provided with some internal and external documentation. Binaries are provided and there is some documentation on how to install the software. There is no useful information on porting. Porting is prohibitively expensive, but some portions (e.g. modules, functions) of the code may be portable.

Level 3 – The software is only portable with significant costs.

The complete source code is available, without external dependencies that are portable, but the software cannot be ported without significant changes to the software or the target context. Documentation on porting the code to another platform or application is missing or deficient. Porting would not be practical or cost effective.

Level 4 – The software may be portable at a reasonable cost.
The cost benefits of using the software slightly outweigh the cost of developing new software. Documentation is barely sufficient, but may contain some useful information on porting to another platform or application. Porting will nonetheless require significant effort. Only at this level is it generally worth considering porting the software.

Level 5 – The software is moderately portable.

The software can be ported with only relatively small changes necessary to the context or the software itself. Documentation on porting exists and is complete, but requires considerable effort and expertise. Some rudimentary understanding of the underlying software or the target system may be necessary.

Level 6 – The software is portable.

The software can be ported to most major systems without modification. The documentation, however, addresses porting to a large number of systems that are identified. Any modifications

needed to port the software to these systems are well described in the documentation and would be relatively easy to implement.

Level 7 – The software is highly portable.

The software can be ported to all but the most obscure or obsolete systems without modification. The documentation is complete and thorough. No changes to the software are necessary and the effort to port the software is minimal.

Level 9 – The software is completely portable.

The software can be ported to all systems since it runs on an application layer rather than on the underlying operating system layer. Such software is written in languages Java, C#, etc. In theory at least, the software will run on any system in which the appropriate application layer has been installed.

Standards Compliance

Level 1 – No standards compliance.

Neither the software nor the software development process adheres to any identified standards other than those inherent in the software languages employed.

Level 2 – No standards compliance beyond best practices.

The software and software development process adhere, at least in part, to some common best practices, but do not identify or claim compliance with any recognized standard.

Level 3 – Some compliance with local standards and best practices.

The software and software development process comply with standards and best practices defined locally by the development organization.

Level 4 – Standards compliance, but incomplete and untested.

The software and software development process attempt to comply with recognized standards, but without verification. Standards compliance is thus untested and may not be complete.

Level 5 – Standards compliance with some testing.

The software and software development process comply with recognized standards, but verification of compliance is incomplete. Standards compliance may not be followed by all components.

Level 6 – Verified standards compliance with proprietary standards.

The software and software development process comply with specific and proprietary standards (such as Windows GUI) and compliance with those standards has been verified through testing.

Level 7 – Verified standards compliance with open standards.

The software and software development process comply with specific open standards and compliance with those standards has been verified through testing.

Level 8 – Verified standards compliance with recognized standards.

The software and software development process comply with internationally recognized standards such as W3C, XML, XHTML, WAI, IP for Web; or ANSI/ISO (C/C++), JCP (Java), for software; and CMMI, IEEE Software Engineering Standards for development process. Standards compliance has been verified through testing, but not by an independent testing organization.

Level 9 – Independently verified standards compliance with recognized standards.

The software and software development process comply with internationally recognized standards. Independent and documented standards compliance verification is included with the software. The development organization maintains standards compliance in its development process through regular testing and certification from an independent group.

Support

Level 1 – No support available.

The original developer of the code is not known, not locatable, or is refusing support.

Level 2 – Minimal support available.

There is known contact information available for the original developer(s) and they are willing to provide minimal, occasional support.

Level 3 – Some support available.

Contact information is available and there is a willingness to provide some support infrequently, without guarantees. This may include things such as providing makefiles or different flavors of the code for different contexts.

Level 4 – Moderate systematic support is available.

Latest updates/patches are usually made available. Support is available, but may be intermittent.

Level 5 – Support provided by an informal user community.

There is an informal user community that provides answers, for example, via a Web site FAQ.

Level 6 – Formal support available.

Support is centralized in a web site containing relevant resources, answers to FAQ, and other useful information.

Level 7 – Organized/defined support by developer available.

There is organized and defined support by the developer with email/telephone help desk and links to case studies and other relevant information. No continuity of support implied.

Level 8 – Support available by the organization that developed the asset.

The support is by an organization and is well defined with frequent updates, releases, etc., and help desk. Continuity of support is implied. Support may be free or fee-based and may be offered by a third party.

Level 9 – Large user community with well-defined support available.

This may include resources such as a help desk, a Web site for the latest information, an active discussion group willing to answer questions, frequent patches and updates as well as consolidation of changes by the community. One example would be the Linux operating system.

Verification and Testing

Level 1 – No testing performed.

Ideas have been translated into software development. Examples might include studies of development languages, prototype, or diagram of interface. Requirements have not been verified, and there is no formal test mechanism in place.

Level 2 – Software application formulated and unit testing performed.

Software application compiles, and executes with known inputs. For example, a prototype application where there is no testing or validation to support the software, but only testing to demonstrate a prototype. Requirements may not be finalized yet, or overall testability of the software determined.

Level 3 – Testing includes testing for error conditions and handling of unknown input.

Software applications have been 'white box' tested. This includes both known and unexpected inputs to the application. This level of testing has been incorporated into the build and/or deployment mechanism.

Level 4 – Software application demonstrated in a laboratory context.

Following successful testing of inputs and outputs, the testing has integrated an application to establish that the "pieces" will work together to achieve concept-enabling levels. This validation has been devised to support the concept that was formulated earlier, and is consistent with the requirements of potential system applications. The validation is relatively "low-fidelity" compared to the eventual system – it could be composed of ad hoc discrete components in a laboratory; for example, an application tested with simulated inputs.

Level 5 – Software application tested and validated in a laboratory context.

The fidelity of the software application testing has not been demonstrated. The software application must be integrated with reasonably realistic supporting elements so that the total application (component level, sub-system level, or system level) can be tested in a "simulated" or somewhat relevant context. At this level, issues such as scalability, load testing, and security are addressed when applicable.

Level 6 – Software application demonstrated in a relevant context.

The fidelity of the software application testing has not been demonstrated. The software application must be integrated with existing elements and interfaces so that the total application (component level, sub-system level, or system level) can be tested and validated in a relevant context. At this level, issues such as number of users and operational scenarios, as well as load testing and security are addressed if applicable.

Level 7 – Software application tested and validated in a relevant context.

The software application testing meets the requirements of the application that apply to the software when it is to be delivered or installed. The software application has been tested in the lab so that the application can be validated as if the software were delivered for use in another context. At this level, all issues have been resolved regarding security and operational scenarios.

Level 8 – Software application "qualified" through test and demonstration (meets requirements) and successfully delivered.

The software has passed testing and meets all requirements of the software, with the additional testing of the software delivery and installation for various applications.

Level 9 – Actual software application tested and validated through successful use of application output.

Demonstrable that for any application of the software, testing shows the software meets all defined requirements.

8. Uses of the RRLs and Topic Area Levels

The WG has considered potential uses of the Reuse Readiness Levels (RRLs), and some of these have been mentioned above and presented before [14]. As metadata for reusable software assets stored in catalogs and repositories, RRLs provide a guide to reusers. The RRLs can help reusers quickly asses the maturity of candidate assets for their reuse efforts, narrowing down the number of possible solutions they must consider in detail. The RRLs and the RRL topic area levels in particular can serve as an indicator of areas to focus on when creating reusable assets, as a guide to providers. The topic areas were selected because they were deemed important factors that contribute to the reusability of software. By assessing their software assets in each of the topic areas, providers can identify the strengths and weaknesses of their assets and work to improve the reusability of the assets using the levels as a guide.

It has also been recognized that RRLs could eventually become part of requests for proposals or contracts, which require a reuse approach or explanation of how assets are being made reusable. Projects could undertake reusability improvement efforts, indicating that software that begins at one RRL will be developed to and released at a higher RRL. By maturing the reusability of the software, the chances of it being reused would be higher, and enable more projects to benefit from this work. Projects that involve new software development could propose to make their resulting work available for reuse and indicate the planned RRL that would be targeted for release of the software. Projects that will be reusing software could indicate the RRLs of the asset(s) being considered for reuse

and demonstrate how this reuse provides benefits to the proposed work (e.g., by reducing development time and costs). The WG is also considering how the RRLs could be used by the upcoming Earth science decadal survey missions (and other similar future missions) and plans to work with the Innovative Partnerships Program Office to pilot a program to include the RRLs in the New Technology Reports that are required as part of NASA software releases.

The RRLs have been developed with different audiences in mind. For example, a project manager may prefer to have a single number that is quick and easy to understand. The summary RRLs serve this purpose. However, software developers may prefer more detailed information, and the RRL topic area levels could serve this purpose. Software may be assessed directly against the summary RRLs or against each of the nine topic areas. In the latter case, an average of the topic area level assessments could serve as the overall RRL summary level, but, in such cases, the individual scores should be preserved. This relates back to the first potential use – as metadata for reusers. If the individual topic area level scores are provided as well as the overall RRL, potential reusers may find the additional information valuable. For example, a reuser could assign weights to the topic area levels (e.g., maybe extensibility is not a concern, as long as the software does what it is supposed to do) then perform a weighted average of candidate software's topic area levels to determine which ones are most suited for his/her needs.

Also, the topic area levels may be used by developers to estimate the RRL of their software, for example, by assigning a value to each topic area and averaging them together to determine an overall RRL (which would be truncated if necessary to produce an integer value). They may also be used to help developers improve the reusability of their software, by providing guidance on how various aspects of the software could be developed further in ways that enable reuse. Providing the topic area level scores in addition to an overall RRL also benefits reusers by providing them with additional information about the reuse readiness of the asset.

The topic area levels also can be used by reusers in estimating which software assets are most suitable for their needs. By assigning a value to each topic area representing the minimum score required for that area and possibly applying a weighting factor to indicate the relative importance of different areas, a weighted average can be calculated that would represent the minimum overall RRL required for an asset to be considered by the reuser. As an example of applying weights, if a reuser was not interested in extending the reusable asset, he/she might give a weight of zero to the extensibility topic area and only average together the other eight topic areas to obtain an overall RRL that has been weighted for the reuser's needs. The WG has prototyped a simple RRL Calculator web page that could be used for these purposes. If individual topic area level scores are provided by the asset developer, these may also be useful in making a decision about what software assets are most likely to meet the reusers' needs.

The potential uses described here are not an exhaustive list. They are meant to provide examples of how the RRLs could be used by different types of users in different situations, and how reuse efforts could benefit from having RRL information available. The WG hopes that as the RRLs become used and established as a valid method for measuring the reuse maturity of software, additional uses for the levels will be found, thus expanding their use and practicality.

9. Summary

A set of nine Reuse Readiness Levels (RRLs) is presented in this document. The RRLs are focused on pinpointing the ability of software components, software systems, and interfaces to be reused in a given context and on pinpointing the potential downstream reusability of software components, systems, and interfaces. Similar to TRLs and their relationship to software maturity, we expect RRLs to become a measure synonymous with a software asset's generality, domain independence, and overall reusability in a software domain.

This document describes the history of the RRLs as they were derived within the NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG). Short summaries and longer descriptions of the RRLs are provided. In addition, nine topic area levels related to reuse (including documentation, support, standards compliance, etc.) were identified as a means to score software assets in terms of relevance and importance to reuse. Those scores can then be weighted and averaged to determine an overall RRL level for a particular software asset.

Uses of the RRLs were also highlighted, including discussion of leveraging the RRLs in New Technology Reports (NTRs) as a means of assessing the reusability of software technology at NASA, as well as a description of the use of RRLs in modern Earth science decadal survey missions, and finally a discussion on a web-based RRL calculator that can be used to interactively explore and compute RRLs for technology.

10. References

1. Technology Readiness Levels: A White Paper, http://www.hq.nasa.gov/office/codeq/trl/trl.pdf

2. Technology Readiness Levels Applied to Software, Appendix II of http://www.nasa.gov/pdf/251088main_2008_SWOY_Sum_Eval_Doc and Defs.pdf

3. Technology Readiness Assessment (TRA) Deskbook, https://acc.dau.mil/CommunityBrowser.aspx?id=18545

4. Technology Readiness Assessments for IT and IT-Enabled Systems, http://www.stsc.hill.af.mil/crosstalk/2005/05/0505Gold.pdf

5. OPEN Process Framework's Technology Readiness Assessment, http://www.opfro.org/Components/WorkProducts/ArchitectureSet/TechnologyReadinessAssessment/TechnologyReadinessAssessment.html

6. Using the Technology Readiness Levels Scale to Support Technology Management in the DoD's ATD/STO Environments (A Findings and Recommendations Report Conducted for Army CECOM), http://www.sei.cmu.edu/reports/02sr027.pdf

7. An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software, http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/09/22680315a.pdf

8. An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software, http://www.sei.cmu.edu/reports/04tr013.pdf

9. Smith, J.D., II. ImpACT: An Alternative to Technology Readiness Levels for Commercial-Off-The-Shelf (COTS) Software. In: COTS-Based Software Systems. LNCS 2959, pp. 127–136. Springer, Berlin/Heidelberg (2004). http://dx.doi.org/10.1007/b96987

10. Business Readiness Rating for Open Source, http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf

11. Marshall, J.J; Olding, S.W.; Wolfe, R.E.; Delnore, V.E., "Software Reuse Within the Earth Science Community," *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on,* pp. 2880–2883, July 31, 2006 – Aug. 4, 2006; http://dx.doi.org/10.1109/IGARSS.2006.740

12. Marshall, J.J.; Downs, R.R., "Reuse Readiness Levels as a Measure of Software Reusability," Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International Conference on, vol. 3, pp. III-1414 – III-1417, 7–11 July 2008; http://dx.doi.org/10.1109/IGARSS.2008.4779626

13. Reuse Readiness Levels (RRLs) – A Work in Progress, http://www.esdswg.org/softwarereuse/Resources/library/6th-esds-wg-meeting/ESDSWG6RRLSlides.pdf/view

14. Reuse Readiness Levels (RRLs): Proposed Topic Area Levels, http://www.esdswg.org/softwarereuse/Resources/library/6th-esds-wg-meeting/ESDSWG6RRLPoster.ppt/view

15. Marshall, J.J., Berrick, S.W., Bertolli, A., Burrows, H., Delnore, V.E., Downs, R.R., Enloe, Y., Falke, S., Folk, M., Gerard, N., Gerard, R., Hunter, M., Jasmin, T., McComas, D., Samadi, S., Sherman, M., Swick, R., Tilmes, C., Wolfe, R.E. (2007), A Community-Developed Measurement of the Reusability of Software Through Reuse Readiness Levels, *Eos Trans. AGU, 88*(52), Fall Meet. Suppl., Abstract IN31A-0074

16. Reuse Readiness Levels (RRLs) – A Work in Progress, http://www.esdswg.org/softwarereuse/Resources/events/event_highlights/2008_Winter_ESIP-RRL-Slides.pdf/view