# Reuse Enablement System (RES) Architecture Study

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

July 23, 2007

Updated February 14, 2008

## Earth Science Data Systems Software Reuse Working Group

Editor:

James Marshall (Innovim / NASA Goddard Space Flight Center)

Contributing Working Group Members:

Angelo Bertolli (Innovim / NASA Goddard Space Flight Center)

Nancy Casey (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center)

Bradford Castalia (University of Arizona)

Victor Delnore * (NASA Langley Research Center)

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

Ryan Gerard (Innovim / NASA Goddard Space Flight Center)

Mary Hunter (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)

Robert Wolfe * (NASA Goddard Space Flight Center)

* Co-chair

Working Group Participants:

Nadine Alameh (MobiLaps LLC)

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

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

Stefan Falke (Washington University in St. Louis)

Michael Folk (National Center for Supercomputing Applications)

Emily Greene (Raytheon Company)

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

Steve Olding (Everware / NASA Goddard Space Flight Center)

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

Fred Watson (California State University, Monterey Bay)

Acknowledgements:

Michael Chyatte (Computer Science Corporation / NASA Goddard Space Flight Center) – SourceMotel site running GForge

Thomas Clune (NASA Goddard Space Flight Center) – SourceMotel site running GForge

Kathy Fontaine (NASA Goddard Space Flight Center) – Earth Science Data System Working Group Coordinator

Gene Major (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center) – Global Change Master Directory

Lola Olsen (NASA Goddard Space Flight Center) – Global Change Master Directory

## Table of Contents

1.0 EXECUTIVE SUMMARY .......... 1

2.0 BACKGROUND .......... 3

3.0 APPLICABLE AND SUPPORTING DOCUMENTS .......... 4

4.0 REQUIREMENTS SUMMARY .......... 5

4.1 DESCRIPTION/PURPOSE OF REQUIREMENTS .......... 5
4.2 STATEMENT OF REQUIREMENTS .......... 6

5.0 REVIEW OF AVAILABLE SOFTWARE PACKAGES .......... 8

5.1 GFORGE .......... 9
5.2 SAVANE .......... 13
5.3 XOOPS .......... 17
5.4 OTHER SOFTWARE PACKAGES INSPECTED .......... 22

6.0 REVIEW OF AVAILABLE SYSTEMS .......... 23

6.1 GLOBAL CHANGE MASTER DIRECTORY (GCMD) .......... 23

7.0 REVIEW SUMMARY .......... 28

8.0 DECISION AND CONCLUSION .......... 30

8.1 NEXT STEP .......... 31

9.0 REFERENCES .......... 31

APPENDIX A – SOFTWARE REUSE QUESTIONNAIRE .......... 32

APPENDIX B – ENABLING SYSTEMS RECOMMENDATION .......... 33

APPENDIX C – GENERAL REQUIREMENTS .......... 34

APPENDIX D – COCOMO 81 DESCRIPTION .......... 36

APPENDIX E – SAVANE GAP ANALYSIS .......... 37

APPENDIX F – XOOPS GAP ANALYSIS .......... 52

APPENDIX G – GLOSSARY OF TERMS .......... 60

# 1.0 Executive Summary

One of the primary goals of the Software Reuse Working Group, one of the NASA Earth Science Data Systems (ESDS) Working Groups, is to save NASA and its partners time and money by driving down the cost and time of system development and by reducing or eliminating expensive redundancy and duplication in system development. To this end, the development of a Reuse Enablement System (RES) is proposed as a way to provide members of the Earth science software development community with easy access to reusable software assets. An architecture study has been conducted to recommend a method for creating the proposed RES. The recommendation offers a low-cost solution: implementing the XOOPS open source content management system, modified as necessary to meet the requirements. Preliminary estimates indicate that this system could be implemented in about 8 person-months of staff time.

As the number of Earth observing instruments and captured data volume increase, so do the complexity and costs associated with software development in support of data transformation, analysis, processing, management, and end-product implementation. Software development costs can be high and the time needed to develop new applications can be considerable. The Earth Observing missions have aided in amplifying knowledge of the Earth system by generating many useful scientific data. To maximize the use of these data, the Earth science community must be able to spend less time, money, and effort on software development and more on scientific work. Reusing software, including open source software, has many benefits such as increased productivity, reduced schedule, and improved quality. However, realizing these benefits for Earth science has been challenging.

Our survey of the Earth science community has indicated that lack of a centralized domain-specific software repository or catalog system addressing the needs of the Earth science community is a major barrier to software reuse within the community. A trade study has been conducted to examine a variety of sites as potential platforms to enable software reuse for the Earth science community. The Reuse Enablement System (RES) Trade Study (dated November 17, 2005) revealed that none of the evaluated repository or catalog systems can adequately satisfy the needs of the community. Based on the results of the trade study, it is recommended that NASA should provide the necessary support for a reuse enablement system dedicated to the Earth science community that could be expanded to include the space science community. In support of the recommendation, technology options for a reuse enablement system have been evaluated and this report on the architecture study has been prepared to offer an expeditious and cost-effective solution for such a system.

A number of software packages and systems were examined for their ability to meet our requirements, as presented in the Reuse Enablement System (RES) Requirements document dated September 18, 2006. This was done through both creating prototypes of systems and examining existing systems that use the same software packages and systems. The results of our study show that using XOOPS with appropriate modifications is the best option for creating a Reuse Enablement System that will provide the community of Earth science software developers with reusable software assets. A basic summary of our results is shown in the following table.

| **Approach Studied** | **# Requirements Met** | **# Requirements Not Met** | **# Requirements Partially Met** | **Development Effort Estimate [staff-months]** |
|---|---|---|---|---|
| XOOPS | 40 | 9 | 5 | 8.12 |
| Savane | 24 | 20 | 10 | 34.01 |
| GCMD | 26 | 24 | 4 | N/A |
| GForge | 20 | 26 | 8 | N/A |

## 2.0 Background

To address the technical issues required to enable and facilitate reuse of software assets within NASA's Earth Science Enterprise (ESE), the Software Reuse Working Group was created as part of the NASA Earth Science Data Systems (ESDS) Working Groups. This was the result of one of the recommendations from the NASA HQ-commissioned Strategic Evolution of ESE Data Systems (SEEDS) Study; the SEEDS activity became the ESDS Working Group activity. The Software Reuse Working Group was chartered to oversee the process that will maximize the reuse potential of such software components in order to: (1) drive down the cost and time of system development and reduce/eliminate unnecessary duplication of effort; (2) increase flexibility and responsiveness relative to Earth science community needs and technological opportunities; and (3) increase effective and accountable community participation.

These objectives, the goals of the Working Group, include:

- recommending and supporting activities that help increase awareness of available software components,
- increasing awareness of the value of reuse, provide needed processes and mechanisms,
- disseminating successful reuse strategies,
- and addressing related intellectual property and policy issues.

In the process of fostering greater software reuse across the Earth science community, a wide variety of approaches have been considered to help meet differing needs and priorities. One such approach has been the creation of the Software Reuse Working Group portal web site (see reference 1 in Section 9). The portal contains information on reusable assets, resources such as events, publications, open source software in general, and funding opportunities, as well as information about activities relevant to reuse. Thus, it provides members of the community with a central location for finding information about software reuse.

The goal of these software reuse activities is to encourage Earth science software developers to make use of existing software assets (including open source software) to provide them with a convenient way to locate and obtain such assets, and to encourage them to develop products for reuse by others. The process of creating a new software product by reusing existing components can be likened to the building of a house. The consumers will be able to buy a completed house, but it is the builders who create the house from a variety of pre-fabricated components such as the frame, windows, and plumbing. Alternatively, consumers may get parts to build their own house, if these parts are well packaged for use directly by the consumers. By using tools, parts, and methods that have been tested over time and are known to work well, it becomes easier and more efficient for them to build the house. Likewise, if software developers can make use of existing software components, it will be easier and more efficient for them to create new products.

Software released under an open source license is publicly available and other software developers can read, modify, and redistribute the source code. Increased use of open source licensing is recommended as an important enabler for software reuse. The licensing mechanism of open source, compared to traditional software licensing, eliminates a significant barrier to code sharing and thus helps to encourage and promote reuse. However, open source licensing is not appropriate for all types of software and traditionally licensed software can still be reused. Therefore, an effective reuse program has to accommodate both open source and non-open source software.

To facilitate the software reuse process, developers need to be able to easily locate and evaluate the available reusable artifacts. These were identified as important factors in a survey (OMB #2700-0117) that was conducted to assess the reuse practices of the Earth science community. See Appendix A for additional information about the survey.[1] The results showed that when people did not reuse software, the primary reasons were because they did not know where to look and they did not know such reusable software existed. In addition, the survey revealed that a catalog or repository for reusable artifacts is the best means of increasing software reuse within the Earth science community. For this reason, the reusable artifacts should be classified and made available through an appropriate reuse enablement system (e.g., libraries, catalogs, repositories) that can facilitate searching and indexing. These systems are an essential ingredient in transforming ad-hoc reuse (which is largely dependent on personal knowledge and word of mouth dissemination of information about the availability of reusable artifacts) to systematic reuse as an integral part of the software development process.

To achieve the above goal, the Software Reuse Working Group was tasked to research and evaluate existing software catalog and repository systems within NASA, specifically the GCMD and the NASA Open Source Agreement site, as possible alternatives to: (1) hosting software assets for the Earth science community and/or (2) developing an Earth science Reuse Enablement System by using existing enablement system reusable infrastructure software components. See Appendix B for the report containing the original recommendation and the response by NASA HQ. These sites as well as other NASA sites and a variety of non-NASA sites performing similar roles were examined and reviewed in a trade study, the results of which showed that none of the existing systems perform the role of providing software developers in the Earth science community with the types of reusable assets they find most useful (see the Reuse Enablement System (RES) Trade Study for more details). It is important to note that this study focuses on the needs of the Earth science community and that the resulting recommendations are not trying to supplant the many accessible open source sites that currently provide registries of general-purpose software. With the need for a new system identified, the study investigated different architectures and methods for creating a Reuse Enablement System that would meet the reuse needs of the Earth science software development community. This document provides an outline of the formalized requirements that were used to evaluate existing architectures (additional details are available in the Reuse Enablement System (RES) Requirements document), the detailed evaluations for a number of software packages and systems, and the results of our study.

## 3.0 Applicable and Supporting Documents

- Reuse Enablement System (RES) Trade Study (November 17, 2005)

  Author: NASA ESDS Software Reuse WG

[1] See also the Proceedings of the 2004 IEEE International Geoscience and Remote Sensing Symposium, vol. 3, pp. 2196-2199, “Strategies for Enabling Software Reuse within the Earth Science Community” by Samadi et al. for preliminary results from an earlier, almost identical survey or the Proceedings of the 2006 IEEE International Geoscience and Remote Sensing Symposium, vol. 6, pp.2880-2883, “Software Reuse Within the Earth Science Community” by Marshall et al. for initial results from the most recent survey.

- Reuse Enablement System (RES) Requirements (September 18, 2006, revised May 7, 2007)

  Author: NASA ESDS Software Reuse WG

- Reuse Enablement System (RES) Use Cases (August 10, 2006)

  Author: NASA ESDS Software Reuse WG

# 4.0 Requirements Summary

## 4.1 Description/Purpose of Requirements

The primary function of a Reuse Enablement System (RES) is to facilitate the distribution and reuse of software development artifacts across the Earth science community, with possible extension to space science. The reusable artifacts supported by the system will include software components and other digital artifacts used in the software development process. It must also have the ability to function as both a catalog and a repository, the main difference between the two being that a catalog stores links to artifacts while a repository stores the actual artifacts themselves. By being able to perform both tasks, the RES provides more options to the user and is able to go beyond what either a catalog or a repository can do alone.

The primary users for the Reuse Enablement System are NASA-funded software developers within the Earth science community. The other category of user is researchers and scientists in various organizations who may be involved with NASA projects. Others are academic scientists or members of research communities. In some cases, the users are also asset providers, implementing software assets and delivering them to the RES for dissemination.

The Software Reuse Working Group conducted a workshop to identify the functional requirements needed for a software Reuse Enablement System (RES) supporting the Earth science community. Several members of the Working Group participated in this workshop and helped draft the initial set of use cases that were developed into requirements. Over a period of several months, these requirements were refined through weekly and monthly telecons and finalized during a review at the October 2004 Data Systems Working Group meeting. The result of this work identified a number of requirements in the following areas: general, search, user registration, asset usage, asset submission, content management, and system administration. These are described in detail in Appendix C.

The general requirements include the kind of features that all systems are expected to exhibit, such as supporting remote access through standard Internet browsers and allowing administrators to generate reports including metrics. It also includes non-functional requirements such as the system being in the Earth science domain, providing the types of assets that are most useful to software developers, and providing a method for appraising the submitted assets for quality control purposes. The focus on the Earth science domain was considered important because it enables the repository to have an asset classification system specific to the needs of the target audience and Earth science assets would not be obscured by large numbers of non-relevant artifacts. The functional requirements can be viewed in the larger categories as noted above (see Appendix C) or in smaller categories based on the specific functions they provide, the use cases for the system.[2] For example, one obvious required function is that the system shall allow users to acquire an asset from the system.

The requirements study was conducted for several months in 2006 to review the proposed descriptive requirements and to create formal requirement statements from these requirements. Several members helped draft the initial statements, which were then refined through weekly and monthly telecons, being finalized at one of the monthly telecons. The formalized requirements are given in the following section.

## 4.2 Statement of Requirements

The following are the titles of the formal requirements for the Reuse Enablement System. See Appendix G for a glossary of terms and additional definitions. Descriptions of the requirements and additional details about the development of these requirements can be found in the Reuse Enablement System (RES) Requirements document.

| Requirement Number and Title |
|---|
| *1 – Users and User Information* |
| *1.1 – Support for User Types* |
| R1.1.1 – Support for Consumer User |
| R1.1.2 – Support for Provider User |
| R1.1.3 – Support for Administrator User |
| R1.1.4 – Support for Content Manager User |
| *1.2 – User Information Storage* |
| R1.2.1 – Storage of Common User Information |
| R1.2.2 – Storage of Provider Information |
| *1.3 – User Interface* |
| R1.3.1 – User Profile Management |
| R1.3.2 – User Request Account Deletion |
| *2 – Asset Storage and Management* |
| *2.1 – Asset Information Storage* |
| R2.1.1 – Storage of Asset Information |
| R2.1.2 – Storage of Asset Resources |
| R2.1.3 – Storage of Asset Versions |
| R2.1.4 – Scanning of Asset Uploads |
| *2.2 – Asset Discovery* |
| R2.2.1 – Display Alphabetical Listing of Assets |
| R2.2.2 – Provide Search for Assets |
| R2.2.3 – Display Hierarchical Navigation of Assets |
| *2.3 – Asset Management* |
| R2.3.1 – Provider Registration of New Assets |
| R2.3.2 – Provider Modification of Assets |

[2] Reference 2 in Section 9 – General WG Documents, Reuse Enablement System (RES), Support and Enablement, Reuse Enablement System Use Cases

| |
|---|
| R2.3.3 – Provider Approval of Asset Modifications |
| R2.3.4 – Provider Request for Asset Removal |
| R2.3.5 – Provider Categorization of Assets |
| *2.4 – Asset Feedback* |
| R2.4.1 – Collection of Comments About Assets |
| R2.4.2 – Collection of Quantitative Feedback |
| R2.4.3 – User Registration of Asset Usage |
| R2.4.4 – Feedback by Contacting Providers |
| R2.4.5 – Display Feedback |
| *2.5 – Asset Metrics and Reports* |
| R2.5.1 – Collect Number of Downloads |
| R2.5.2 – Collect Number of External Links Accessed |
| R2.5.3 – Collect Number of Registered Users for Assets |
| R2.5.4 –Summarize Ratings from Quantitative Feedback |
| 2.6 – Asset Access Control |
| R2.6.1 – Limit Access of Certain Users from Certain Assets |
| *3 – Send and Manage Notifications* |
| *3.1 – Send Notifications for Asset Events* |
| R3.1.1 – Send Notification on Modification of Asset |
| R3.1.2 – Send Notification on Submission of New Feedback |
| *3.2 – Send Notifications for System Events* |
| R3.2.1 – Send Administrative Notification for Asset Information |
| R3.2.2 – Send Administrative Notification for System Information |
| *3.2 – Notification Management* |
| R3.2.1 – User Addition of Notifications for Assets |
| R3.2.2 – User Removal of Notifications |
| *4 – System Operations* |
| *4.1 – System Feedback* |
| R4.1.1 – Collection of System Problems |
| R4.1.2 – Collection of Suggestions |
| R4.1.3 – Feedback by Contacting Administrators |
| *4.2 – System Policies Compliance, Security, and Privacy* |
| R4.2.1 – Verification of Provider Information |
| R4.2.2 –Verification of  Provider through Secondary Method or Contact |
| R4.2.3 – Security of Sensitive Transmitted Information |
| R4.2.4 – Security of Stored Information |
| R4.2.5 – Deletion of Users for Policy Enforcement |
| R4.2.6 – Protection of Private Information |
| R4.2.7 – Compliance with other Technical, Accessibility, and Security Requirements |
| R4.2.8 – Policies Availability to Users |
| *4.3 – Repository and Catalog* |
| R4.3.1 – Function as a Repository |
| R4.3.2 – Function as a Catalog |
| R4.3.3 – Selection of System Behavior by Provider |

| |
|---|
| R4.3.4 – Enforcement of Asset Storage Limit |
| *4.4 – Asset Cleanup* |
| R4.4.1 – Asset Deprecation by Content Managers |
| R4.4.2 – Asset Removal by Administrators |
| *4.5 – Data Integrity* |
| R4.5.1 – Verification of Data by Providers |

## 5.0 Review of Available Software Packages

We researched catalog and repository system software packages as an option for building a new Reuse Enablement System (RES). This would allow us to reuse an existing software package that provides most of our desired functionality as the foundation of the RES. Modifications would be made as necessary to ensure that the RES meets all of the stated formal requirements. The packages that were evaluated and reviewed as part of this study include:

1) GForge (see reference 3 in Section 9)
2) Savane (see reference 4 in Section 9)
3) XOOPS (see reference 5 in Section 9)

Some other packages were examined, but not reviewed in detail. A brief description of each of the sites in the following list follows the detailed review of the sites listed above.

4) Fedora Digital Repository System (see reference 6 in Section 9)
5) JBoss Portal (see reference 7 in Section 9)
6) Liferay Portal (see reference 8 in Section 9)
7) Repository in a Box (see reference 9 in Section 9)

This section describes the result of our study on how well these software packages and existing systems meet the formal requirements defined in the previous section. The review includes sections on the installation of the package/system, how well it meets our requirements, a gap analysis of what would be necessary to meet any of our requirements that the package/system does not currently meet, and maintenance and support.

As part of our evaluation, we include a level of effort estimate in the gap analysis section for the requirements that the systems or packages do not meet or only partially meet. The development effort is based on Barry Boehm's original Constructive Cost Model (COCOMO), the details of which are provided in Appendix D. The software is generally considered semi-detached software because the code being modified is somewhat modular. The software must operate within (is partially embedded in) an open-source software application that is not considered overly complex. Because of this we have assigned COCOMO estimates according to complexity:

| Complexity | Classification |
|---|---|
| 2 – 3 | Organic |
| 4 – 9 | Semi-embedded |
| 10 | Embedded |

## 5.1 GForge

GForge is a collaborative development tool based on the SourceForge code. It is a fork of SourceForge code version 2.61, and at the time of writing is currently at version 4.5. Some of the GForge features include message forums, mailing lists, source code management repositories like CVS and Subversion, access control to repositories, role-based access controls, document management with approval queue, and command line interface.

This review of GForge is based primarily on the SourceMotel system (see reference 10 in Section 9) at the NASA Goddard Space Flight Center. This web site runs a customized version of the GForge software version 4.0. We received a demonstration of the site from an experienced user of the system and talked with one of the system administrators about how the site was created and configured and how it is currently maintained.

### 5.1.1 Installation

The installation requirements for GForge, as outlined in the online manual, are as follows:

1. Linux operating system
2. PostgreSQL 7.3 or later
3. Apache 1.3.22 or later
4. openssl 0.9.4 or later
5. mod_ssl 2.4.10 or later
6. PHP 4.0.4 or later built with command line interface support
7. php-pgsql
8. php-mbstring

Additional software packages are listed as optional and include, for example, a PHP accelerator (highly recommended by GForge) and GNU Mailman and Python for mailing list support. The manual also indicates that GForge can be installed on Debian systems using the apt-get command, and there are RPM packages available for installation on RPM-based systems such as

Fedora Core and Red Hat Enterprise Linux. The use of installation packages greatly simplifies the installation process.

The latest version of GForge is 4.5, and according to the SourceMotel staff, it is a good version. It combines PHP, static HTML, and a number of other packages (such as CVS, Mailman, and PostgreSQL) together to provide a collaborative development environment. It is not entirely database-driven however, as it needs to create a new UNIX user account. Also, it integrates with many different components across the system and takes control over a variety of tasks, performing them automatically. This means that GForge cannot easily be run on a system that is being shared and used for multiple purposes; it needs to have its own system. System events are typically strongly tied to CVS. GForge tends to be built for the latest (unstable) version of the Debian Linux operating system, but runs on any Linux operating system.

### 5.1.2 Meeting Requirements

The following table indicates how well GForge meets our stated requirements. Explanatory notes follow, when applicable.

**Table 1 – GForge Requirements Matching**

| **Requirement Number and Title** | **Meets Requirement?** |
|---|---|
| *1 – Users and User Information* | |
| *1.1 – Support for User Types* | |
| R1.1.1 – Support for Consumer User | YES |
| R1.1.2 – Support for Provider User | YES |
| R1.1.3 – Support for Administrator User | PARTIAL |
| R1.1.4 – Support for Content Manager User | NO |
| *1.2 – User Information Storage* | |
| R1.2.1 – Storage of Common User Information | YES |
| R1.2.2 – Storage of Provider Information | NO |
| *1.3 – User Interface* | |
| R1.3.1 – User Profile Management | YES |
| R1.3.2 – User Request Account Deletion | NO |
| *2 – Asset Storage and Management* | |
| *2.1 – Asset Information Storage* | |
| R2.1.1 – Storage of Asset Information | PARTIAL |
| R2.1.2 – Storage of Asset Resources | PARTIAL |
| R2.1.3 – Storage of Asset Versions | YES |
| R2.1.4 – Scanning of Asset Uploads | NO |
| *2.2 – Asset Discovery* | |
| R2.2.1 – Display Alphabetical Listing of Assets | NO |
| R2.2.2 – Provide Search for Assets | YES |
| R2.2.3 – Display Hierarchical Navigation of Assets | YES |
| *2.3 – Asset Management* | |
| R2.3.1 – Provider Registration of New Assets | YES |

| **Requirement Number and Title** | **Meets Requirement?** |
|---|---|
| R2.3.2 – Provider Modification of Assets | YES |
| R2.3.3 – Provider Approval of Asset Modifications | PARTIAL |
| R2.3.4 – Provider Request for Asset Removal | YES |
| R2.3.5 – Provider Categorization of Assets | YES |
| *2.4 – Asset Feedback* | |
| R2.4.1 – Collection of Comments About Assets | NO |
| R2.4.2 – Collection of Quantitative Feedback | NO |
| R2.4.3 – User Registration of Asset Usage | NO |
| R2.4.4 – Feedback by Contacting Providers | PARTIAL |
| R2.4.5 – Display Feedback | YES |
| *2.5 – Asset Metrics and Reports* | |
| R2.5.1 – Collect Number of Downloads | NO |
| R2.5.2 – Collect Number of External Links Accessed | NO |
| R2.5.3 – Collect Number of Registered Users for Assets | NO |
| R2.5.4 –Summarize Ratings from Quantitative Feedback | NO |
| 2.6 – Asset Access Control | |
| R2.6.1 – Limit Access of Certain Users from Certain Assets | NO |
| *3 – Send and Manage Notifications* | |
| *3.1 – Send Notifications for Asset Events* | |
| R3.1.1 – Send Notification on Modification of Asset | NO |
| R3.1.2 – Send Notification on Submission of New Feedback | NO |
| *3.2 – Send Notifications for System Events* | |
| R3.2.1 – Send Administrative Notification for Asset Information | PARTIAL |
| R3.2.2 – Send Administrative Notification for System Information | YES |
| *3.2 – Notification Management* | |
| R3.2.1 – User Addition of Notifications for Assets | NO |
| R3.2.2 – User Removal of Notifications | NO |
| *4 – System Operations* | |
| *4.1 – System Feedback* | |
| R4.1.1 – Collection of System Problems | YES |
| R4.1.2 – Collection of Suggestions | YES |
| R4.1.3 – Feedback by Contacting Administrators | YES |
| *4.2 – System Policies Compliance, Security, and Privacy* | |
| R4.2.1 – Verification of Provider Information | NO |
| R4.2.2 –Verification of  Provider through Secondary Method or Contact | NO |
| R4.2.3 – Security of Sensitive Transmitted Information | YES |
| R4.2.4 – Security of Stored Information | YES |
| R4.2.5 – Deletion of Users for Policy Enforcement | NO |
| R4.2.6 – Protection of Private Information | YES |
| R4.2.7 – Compliance with other Technical, Accessibility, and Security Requirements | PARTIAL |
| R4.2.8 – Policies Availability to Users | NO |

| Requirement Number and Title | Meets Requirement? |
|---|---|
| *4.3 – Repository and Catalog* | |
| R4.3.1 – Function as a Repository | YES |
| R4.3.2 – Function as a Catalog | PARTIAL |
| R4.3.3 – Selection of System Behavior by Provider | NO |
| R4.3.4 – Enforcement of Asset Storage Limit | NO |
| *4.4 – Asset Cleanup* | |
| R4.4.1 – Asset Deprecation by Content Managers | NO |
| R4.4.2 – Asset Removal by Administrators | NO |
| *4.5 – Data Integrity* | |
| R4.5.1 – Verification of Data by Providers | NO |

### 5.1.3 Gap Analysis

The major task here is likely to be removing features that we do not require and do not wish to offer. However, some customization would be necessary to make a GForge system meet our requirements. New user accounts are approved automatically by default; this needs to be changed. This is one of the modifications the SourceMotel staff made to their system. Automatically mirroring posts made in the web forums to the e-mail mailing lists would be very difficult. Incorporating a feedback system would be difficult as well because it needs to work with the database. GForge met the smallest number of requirements of the systems we examined, so we did not do a detailed gap analysis since it was clear that other systems would be better choices.

### 5.1.4 Maintenance and Support

Since GForge is really a collection of smaller software packages joined together, the ease of upgrades depends at least partially on the packages that compose the GForge system. Testing upgrades can be difficult, and it can take a lot of time to understand how the new version operates. For these reasons and because frequent downtime is necessary for configuration, the SourceMotel staff suggested that two systems be used with GForge – one as the operational system and the second as the development system. This is how they operate their system. Upgrades and other modifications are tested in the development system until the staff is sure they work properly, and then the changes are made in the operational system.

The SourceMotel staff also noted some other difficulties in maintaining and supporting the system. In order to change machines, the name of the host must be changed in multiple places. This causes some confusion and makes it difficult to ensure that the host change has been completed properly. With sufficient modifications, the SourceMotel staff solved this issue, but it did cause problems for them. Users and projects cannot be deleted through the standard GForge system interface, causing a problem if user names or project names are to be recycled and reused. Users can be suspended, but projects are essentially permanent. It may be possible to delete users and/or projects by other means, e.g., direct manipulation of the database, but even this may not guarantee success because of the tight integration of GForge with the system. Since GForge takes over tasks such as maintenance of the system's password file, deleting users manually can result in GForge recreating the user account based on information in its database, in an attempt to keep itself synchronized with the host system. Many cron jobs are used to execute tasks at specific intervals, and this timing is critical to proper execution. However, the behavior is undefined, and there is no obvious logic behind the timing settings. The SourceMotel staff learned through experience that modifying the cron job timings can break the GForge system, so they must be left as-is in order for the system to function. Currently, Subversion is not supported, but there are plans to provide support for it in the future. Also, the unique naming conventions of GForge can create problems where, for example, new mailing lists created in Mailman may end up colliding with existing lists or be named non-intuitively.

An additional point to note is that GForge offers professional support for a fee, and the SourceMotel staff did make use of this. However, they also ran into complications that limited the amount of help they were able to obtain from the GForge staff. It was difficult to get both sides to agree on how things should be done, which resulted in an “all-or-nothing” scenario where the GForge staff really only wanted to do things themselves their way.

### 5.1.5 Summary

The GForge system provides a good environment for collaborative development, but it is also a very fragile system. It is relatively easy to break, and it is not always clear why things work the way they do. The SourceMotel staff’s opinion was that GForge should be used only if we plan to use most of the features it provides. Out of our 54 requirements, GForge meets 20, partially meets 8, and does not meet 26.

## 5.2 Savane

Savane is a Web-based Libre Software hosting system based on the SourceForge software. It was originally designed to be an installation of the SourceForge 2.0 software, but became its own package after the SourceForge software became proprietary. Savane provides a collaborative development environment, and we tested version 1.4.

### 5.2.1 Installation

The installation requirements for Savane, as provided in the installation package, are as follows:

1. Apache 1.3.x or greater
2. Perl 5.6 or greater
3. PHP 4.1.0 or greater
4. MySQL 3.x or greater

Some additional GNU/Linux utilities including exim or sendmail are required, and some Perl modules must also be installed.

Savane should be installed on its own server or in a virtual server environment because it makes use of ftp, ssh, cvs/subversion, and other software that require it to manipulate the system users and groups.

The following is a description of the installation and configuration of Savane 1.4.

1. Configure and make the scripts for installation. This first step checks for software dependencies and allows you to set up basic definitions such as Savane's configuration, library, and binary directories.
2. Make the essential parts of Savane.

3. Make the database by using a db admin account, and create a db user for Savane to use.

4. Make the Savane configuration files by answering a series of questions on how you want Savane to operate

5. Install Savane by creating a savane cron job and logrotate job.

### 5.2.2 Meeting Requirements

The following table indicates how well Savane meets our stated requirements. Explanatory notes follow, when applicable.

**Table 2 – Savane Requirements Matching**

| Requirement Number and Title | Meets Requirement? |
|---|---|
| *1 – Users and User Information* | |
| *1.1 – Support for User Types* | |
| R1.1.1 – Support for Consumer User | YES |
| R1.1.2 – Support for Provider User | YES |
| R1.1.3 – Support for Administrator User | PARTIAL |
| R1.1.4 – Support for Content Manager User | PARTIAL |
| *1.2 – User Information Storage* | |
| R1.2.1 – Storage of Common User Information | YES |
| R1.2.2 – Storage of Provider Information | NO |
| *1.3 – User Interface* | |
| R1.3.1 – User Profile Management | YES |
| R1.3.2 – User Request Account Deletion | YES |
| *2 – Asset Storage and Management* | |
| *2.1 – Asset Information Storage* | |
| R2.1.1 – Storage of Asset Information | PARTIAL |
| R2.1.2 – Storage of Asset Resources | YES |
| R2.1.3 – Storage of Asset Versions | YES |
| R2.1.4 – Scanning of Asset Uploads | NO |
| *2.2 – Asset Discovery* | |
| R2.2.1 – Display Alphabetical Listing of Assets | PARTIAL |
| R2.2.2 – Provide Search for Assets | YES |
| R2.2.3 – Display Hierarchical Navigation of Assets | PARTIAL |
| *2.3 – Asset Management* | |
| R2.3.1 – Provider Registration of New Assets | YES |
| R2.3.2 – Provider Modification of Assets | YES |
| R2.3.3 – Provider Approval of Asset Modifications | PARTIAL |
| R2.3.4 – Provider Request for Asset Removal | YES |
| R2.3.5 – Provider Categorization of Assets | NO |
| *2.4 – Asset Feedback* | |

| Requirement Number and Title | Meets Requirement? |
|---|---|
| R2.4.1 – Collection of Comments About Assets | NO |
| R2.4.2 – Collection of Quantitative Feedback | NO |
| R2.4.3 – User Registration of Asset Usage | NO |
| R2.4.4 – Feedback by Contacting Providers | PARTIAL |
| R2.4.5 – Display Feedback | YES |
| *2.5 – Asset Metrics and Reports* | |
| R2.5.1 – Collect Number of Downloads | NO |
| R2.5.2 – Collect Number of External Links Accessed | NO |
| R2.5.3 – Collect Number of Registered Users for Assets | NO |
| R2.5.4 –Summarize Ratings from Quantitative Feedback | NO |
| 2.6 – Asset Access Control | |
| R2.6.1 – Limit Access of Certain Users from Certain Assets | NO |
| *3 – Send and Manage Notifications* | |
| *3.1 – Send Notifications for Asset Events* | |
| R3.1.1 – Send Notification on Modification of Asset | NO |
| R3.1.2 – Send Notification on Submission of New Feedback | NO |
| *3.2 – Send Notifications for System Events* | |
| R3.2.1 – Send Administrative Notification for Asset Information | PARTIAL |
| R3.2.2 – Send Administrative Notification for System Information | PARTIAL |
| *3.2 – Notification Management* | |
| R3.2.1 – User Addition of Notifications for Assets | NO |
| R3.2.2 – User Removal of Notifications | NO |
| *4 – System Operations* | |
| *4.1 – System Feedback* | |
| R4.1.1 – Collection of System Problems | YES |
| R4.1.2 – Collection of Suggestions | YES |
| R4.1.3 – Feedback by Contacting Administrators | YES |
| *4.2 – System Policies Compliance, Security, and Privacy* | |
| R4.2.1 – Verification of Provider Information | NO |
| R4.2.2 –Verification of  Provider through Secondary Method or Contact | NO |
| R4.2.3 – Security of Sensitive Transmitted Information | YES |
| R4.2.4 – Security of Stored Information | YES |
| R4.2.5 – Deletion of Users for Policy Enforcement | YES |
| R4.2.6 – Protection of Private Information | YES |
| R4.2.7 – Compliance with other Technical, Accessibility, and Security Requirements | PARTIAL |
| R4.2.8 – Policies Availability to Users | NO |
| *4.3 – Repository and Catalog* | |
| R4.3.1 – Function as a Repository | YES |
| R4.3.2 – Function as a Catalog | YES |
| R4.3.3 – Selection of System Behavior by Provider | YES |
| R4.3.4 – Enforcement of Asset Storage Limit | NO |

| Requirement Number and Title | Meets Requirement? |
|---|---|
| *4.4 – Asset Cleanup* | |
| R4.4.1 – Asset Deprecation by Content Managers | NO |
| R4.4.2 – Asset Removal by Administrators | YES |
| *4.5 – Data Integrity* | |
| R4.5.1 – Verification of Data by Providers | YES |

### 5.2.3 Gap Analysis

A full description of the gap analysis for Savane can be found in Appendix E. A summary of the analysis appears in the following section, Development Effort.

### 5.2.4 Development Effort

Using the complexity as a measure of the classification, along with our estimate for the number of lines of code necessary to make Savane meet each of our unmet or partially met requirements, we estimated the development effort for each component according to the equations in Appendix D and the complexity/classification table at the beginning of this section. The results are shown in the following table.

| Requirement | Lines of Code | Complexity | Effort |
|---|---|---|---|
| R1.1.3 | 550 | 5 | 1.54 |
| R1.1.4 | 1600 | 9 | 6.33 |
| R1.2.2 | 150 | 5 | 0.36 |
| R2.1.1 | 650 | 6 | 1.85 |
| R2.1.4 | 275 | 10 | 0.76 |
| R2.2.1 | 5 | 3 | 0.01 |
| R2.2.3 | 650 | 6 | 1.85 |
| R2.3.3 | 1150 | 10 | 4.26 |
| R2.3.5 | 510 | 5 | 1.41 |
| R2.4.1 | 550 | 5 | 1.54 |
| R2.4.2 | 510 | 5 | 1.41 |
| R2.4.3 | 110 | 5 | 0.25 |
| R2.4.4 | 60 | 5 | 0.13 |
| R2.5.1 | 510 | 5 | 1.41 |
| R2.5.2 | 510 | 5 | 1.41 |
| R2.5.3 | 500 | 3 | 1.16 |
| R2.5.4 | 500 | 3 | 1.16 |
| R2.6.1 | 600 | 5 | 1.69 |
| R3.1.1 | 100 | 3 | 0.21 |
| R3.1.2 | 100 | 3 | 0.21 |
| R3.2.1 | 100 | 3 | 0.21 |
| R3.2.2 | 100 | 3 | 0.21 |
| R3.3.1 | 600 | 5 | 1.69 |
| R3.3.2 | 100 | 3 | 0.21 |
| R4.2.1 | 600 | 6 | 1.69 |

| Requirement | Lines of Code | Complexity | Effort |
|---|---|---|---|
| R4.2.2 | 110 | 5 | 0.25 |
| R4.2.7 | 150 | 3 | 0.33 |
| R4.2.8 | 100 | 2 | 0.21 |
| R4.3.4 | 10 | 4 | 0.02 |
| R4.4.1 | 105 | 5 | 0.24 |
| **Total Development Effort [staff-months]** | | | 34.01 |
| **Total Development Effort [staff-years]** | | | 2.83 |

Our estimates indicate that it would take approximately 2.83 staff-years of development effort to modify Savane to meet our requirements.

### 5.2.5 Maintenance and Support

We determined that the maintenance effort for Savane was similar to the complexity because Savane does not provide an API for creating modules, or other ways of separating our changes from their code. This means a large effort must be taken to incorporate our changes as future versions of Savane are released. Therefore, we based the level of maintenance on the complexity, using our own judgment to modify the score.

### 5.2.6 Summary

Savane is a complex system that would require many changes to meet our requirements. The level of maintenance required is also relatively high, since it is difficult, if not impossible, to separate modifications we would make from the base code of the system. Out of our 54 requirements, Savane meets 24, partially meets 10, and does not meet 20.

## 5.3 XOOPS

XOOPS is an acronym for eXtensible Object Oriented Portal System. It is a Content Management System (CMS) written in PHP, and it uses the MySQL relational database. It provides a web-based CMS, and its basic functions can be modified through the use of modules provided with the package or downloaded from the Module Repository at the official web site. We used version 2.0.13.2, released on Oct. 28, 2005 for our tests; this was the most recent stable release at the time we performed our evaluations. The current latest stable release is version 2.0.16.

### 5.3.1 Installation

The installation requirements for XOOPS, as outlined in the install wizard, are as follows:

1. WWW server (Apache, IIS, Roxen, etc.)
2. PHP 4.1.0 and higher (4.1.1 or higher recommended)
3. MySQL Database 3.23.XX

Besides installing these programs properly and providing access to a database, there is little that needs to be done. The package's "html" directory contents must be copied to a web-accessible directory on the system in order to install and create the new XOOPS site. When running the installation wizard, it will alert you to the few additional steps that need to be performed prior to installation; these are easy to complete.

Installation of XOOPS is very simple. Pointing a web browser to the “html” directory of the package begins the installation wizard, and following the on-screen instructions performs the installation. If there are any problems during the process, the wizard will point them out. Once they are corrected, the installation process may be resumed or restarted. In some cases, if the installation was halted after some database tables were created, those tables must be removed before restarting the installation, otherwise it will halt again because it cannot create (or overwrite) the existing tables. Completion of the installation process provides a basic XOOPS site, but little functionality. A number of modules are provided in the XOOPS package, and they can be installed easily to provide a more functional web site. We did this, and our analysis is based primarily on this installation. Additional modules that may provide additional functionality are available at the official web site’s module repository.

We encountered no real trouble with the installation process of the system or provided modules. However, one point worth noting is that the URL of the XOOPS site is initially set during the installation process. Changing this may not be a very simple matter however, so some difficulties could arise in moving a XOOPS site to a new web address. Most of the setup time was spent configuring the system and modules because there are many options to understand and settings to select.

### 5.3.2 Meeting Requirements

The following analysis is based on the functionality provided by the default modules included in the installation package. Additional functionality may be provided by other modules in the repository at the official site. If it is known that additional modules can provide functionality that the default ones cannot, a note will be made of this.

The following table indicates how well XOOPS meets our stated requirements. Explanatory notes follow, when applicable.

**Table 3 – XOOPS Requirements Matching**

| Requirement Number and Title | Meets Requirement? |
|---|---|
| *1 – Users and User Information* | |
| *1.1 – Support for User Types* | |
| R1.1.1 – Support for Consumer User | YES |
| R1.1.2 – Support for Provider User | YES |
| R1.1.3 – Support for Administrator User | YES |
| R1.1.4 – Support for Content Manager User | YES |
| *1.2 – User Information Storage* | |
| R1.2.1 – Storage of Common User Information | YES |
| R1.2.2 – Storage of Provider Information | YES |
| *1.3 – User Interface* | |
| R1.3.1 – User Profile Management | YES |
| R1.3.2 – User Request Account Deletion | YES |
| *2 – Asset Storage and Management* | |

| **Requirement Number and Title** | **Meets Requirement?** |
|---|---|
| *2.1 – Asset Information Storage* | |
| R2.1.1 – Storage of Asset Information | YES |
| R2.1.2 – Storage of Asset Resources | YES |
| R2.1.3 – Storage of Asset Versions | YES |
| R2.1.4 – Scanning of Asset Uploads | NO |
| *2.2 – Asset Discovery* | |
| R2.2.1 – Display Alphabetical Listing of Assets | NO |
| R2.2.2 – Provide Search for Assets | YES |
| R2.2.3 – Display Hierarchical Navigation of Assets | YES |
| *2.3 – Asset Management* | |
| R2.3.1 – Provider Registration of New Assets | YES |
| R2.3.2 – Provider Modification of Assets | YES |
| R2.3.3 – Provider Approval of Asset Modifications | PARTIAL |
| R2.3.4 – Provider Request for Asset Removal | YES |
| R2.3.5 – Provider Categorization of Assets | YES |
| *2.4 – Asset Feedback* | |
| R2.4.1 – Collection of Comments About Assets | YES |
| R2.4.2 – Collection of Quantitative Feedback | YES |
| R2.4.3 – User Registration of Asset Usage | NO |
| R2.4.4 – Feedback by Contacting Providers | PARTIAL |
| R2.4.5 – Display Feedback | YES |
| *2.5 – Asset Metrics and Reports* | |
| R2.5.1 – Collect Number of Downloads | YES |
| R2.5.2 – Collect Number of External Links Accessed | YES |
| R2.5.3 – Collect Number of Registered Users for Assets | NO |
| R2.5.4 –Summarize Ratings from Quantitative Feedback | YES |
| 2.6 – Asset Access Control | |
| R2.6.1 – Limit Access of Certain Users from Certain Assets | YES |
| *3 – Send and Manage Notifications* | |
| *3.1 – Send Notifications for Asset Events* | |
| R3.1.1 – Send Notification on Modification of Asset | YES |
| R3.1.2 – Send Notification on Submission of New Feedback | YES |
| *3.2 – Send Notifications for System Events* | |
| R3.2.1 – Send Administrative Notification for Asset Information | YES |
| R3.2.2 – Send Administrative Notification for System Information | YES |
| *3.2 – Notification Management* | |
| R3.2.1 – User Addition of Notifications for Assets | YES |
| R3.2.2 – User Removal of Notifications | YES |
| *4 – System Operations* | |
| *4.1 – System Feedback* | |
| R4.1.1 – Collection of System Problems | YES |

| **Requirement Number and Title** | **Meets Requirement?** |
|---|---|
| R4.1.2 – Collection of Suggestions | YES |
| R4.1.3 – Feedback by Contacting Administrators | YES |
| *4.2 – System Policies Compliance, Security, and Privacy* | |
| R4.2.1 – Verification of Provider Information | YES |
| R4.2.2 –Verification of Provider through Secondary Method or Contact | NO |
| R4.2.3 – Security of Sensitive Transmitted Information | PARTIAL |
| R4.2.4 – Security of Stored Information | YES |
| R4.2.5 – Deletion of Users for Policy Enforcement | YES |
| R4.2.6 – Protection of Private Information | YES |
| R4.2.7 – Compliance with other Technical, Accessibility, and Security Requirements | PARTIAL |
| R4.2.8 – Policies Availability to Users | NO |
| *4.3 – Repository and Catalog* | |
| R4.3.1 – Function as a Repository | NO |
| R4.3.2 – Function as a Catalog | YES |
| R4.3.3 – Selection of System Behavior by Provider | NO |
| R4.3.4 – Enforcement of Asset Storage Limit | YES |
| *4.4 – Asset Cleanup* | |
| R4.4.1 – Asset Deprecation by Content Managers | NO |
| R4.4.2 – Asset Removal by Administrators | YES |
| *4.5 – Data Integrity* | |
| R4.5.1 –Verification of Data by Providers | PARTIAL |

Requirements 2.2.1, 4.3.1, and 4.3.3 can be met through the use of another existing module. For example, the module called PD Downloads that is available on the XOOPS official module repository provides all of the functions necessary to meet these three requirements. It provides an option to browse assets by alphabetical listing and it allows users to upload assets to be stored on the system, which allows the Provider to choose how his/her asset is stored (locally as for a repository, or remotely as for a catalog).

Requirement 1.2.2, Storage of Provider Information, can be met easily by using some of the pre-defined user profile information slots even though they are not specifically designated as organization and area of expertise.

Requirement 2.3.4, Provider Request for Asset Removal, is met through Requirement 4.1.3, Feedback by Contacting Administrators. Although there is no specific mechanism for requesting the removal of an asset, Providers can always contact Administrators to make such a request.

Requirements 4.1.1, Collection of System Problems, and 4.1.2, Collection of System Suggestions, are met through Requirement 4.1.3, Feedback by Contacting Administrators. The only specific feature for reporting bug or sending suggestions is a method for reporting broken links and download files, but since the Administrator can always be contacted, these requirements are all met.

Requirement 4.2.1, Verification of Provider Information, is met because the system can be configured to avoid automatically accepting new registrations, thereby giving Administrators time to complete the verification process.

### 5.3.3 Gap Analysis

A full description of the gap analysis for XOOPS can be found in Appendix F. A summary of the analysis appears in the following section, Development Effort.

### 5.3.4 Development Effort

Using the complexity as a measure of the classification, along with our estimate for the number of lines of code necessary to make XOOPS meet each of our unmet or partially met requirements, we estimated the development effort for each component according to the equations in Appendix D and the complexity/classification table at the beginning of this section. The results are shown in the following table.

| Requirement | Lines of Code | Complexity | Effort |
|---|---|---|---|
| R2.1.4 | 315 | 4 | 0.82 |
| R2.2.1 | 10 | 2 | 0.02 |
| R2.3.3 | 1050 | 3 | 2.53 |
| R2.4.3 | 510 | 3 | 1.18 |
| R2.4.4 | 210 | 3 | 0.47 |
| R2.5.3 | 500 | 1 | 1.16 |
| R4.2.2 | 210 | 3 | 0.47 |
| R4.2.3 | 10 | 1 | 0.02 |
| R4.2.7 | 150 | 2 | 0.33 |
| R4.2.8 | 100 | 1 | 0.21 |
| R4.3.1 | 10 | 1 | 0.02 |
| R4.3.3 | 100 | 1 | 0.21 |
| R4.4.1 | 105 | 3 | 0.23 |
| R4.5.1 | 205 | 3 | 0.45 |
| **Total Development Effort [staff-months]** | | | 8.12 |
| **Total Development Effort [staff-years]** | | | 0.68 |

Our estimates indicate that it would take approximately 0.68 staff-years of development effort to modify XOOPS to meet our requirements.

### 5.3.5 Maintenance and Support

We determined that the maintenance effort for XOOPS was less costly than the complexity because XOOPS provides an API for creating modules. This means our code is more isolated, and can be maintained separately through upgrades of XOOPS. The only time the maintenance may be high is if XOOPS modifies the API, but that is not taken into consideration for the estimates here.

The modular nature of XOOPS allows for easier maintenance and support. If a problem arises, it should be traceable to one particular module that is independent of the rest. This allows the site to maintain its integrity and not lose much functionality while the problem(s) with one module are being solved. It also allows updating of individual components of the system as modules are updated and upgraded. The modules can be maintained separately from the XOOPS core system itself.

Support for XOOPS is good, with the official site operating a forum for support, where users can assist each other. This is separated into topic areas, including general, community support, modules support, themes and templates support, and XOOPS development, each of which has a number of sub-areas for particular types of problems and support.

### 5.3.6 Summary

XOOPS is able to meet most of our requirements with its basic installation and the default modules provided with the package. A few requirements that are not met by the default modules can be met by installing one or more other modules from the module repository on the official site. Some requirements are considered partially met as they have no dedicated feature, but can be implemented by means already available on the system. Together, these factors reduce the amount of customization necessary to make a XOOPS system meet all of our stated requirements. Also, the modular structure of XOOPS simplifies maintenance since code changes can be isolated in their own modules independent of the base code of the system. Out of our 54 requirements, XOOPS meets 40, partially meets 5, and does not meet 9.

## 5.4 Other Software Packages Inspected

There are other software packages for creating software catalogs and repositories in addition to the ones reviewed here. Some of them provide very similar functionality; for example, most Content Management Systems will function in much the same was as XOOPS does. We could not evaluate every possible package in detail, so chose some representative examples. We also received some feedback from members of the Earth science community about other packages we had not seen previously. We considered these options as well, but found that they were generally not suitable for our needs. This section provides a brief description of these packages and the reasons they were not examined in detail. In all cases, time constraints were an issue. We did not have time to implement prototypes of all systems we examined, so we were forced to choose only a few for prototyping. The other packages listed here may be suitable, but on our initial examination, they did not appear to be as simple to create or use as the ones reviewed above.

The Fedora Digital Repository System is open source software repository system developed by Cornell University Information Science and the University of Virginia Library. It is designed to manage digital content, local or remote, through the use of a digital object which describes the objects. All functions are web services and have fine-grained control access policies. The Fedora web site lists library collections management, multimedia authoring systems, archival repositories, institutional repositories, and digital libraries for education as some applications and domains where Fedora has been found useful. We examined existing instances of this system, and found that there were no user accounts, which is an important requirement for our desired system. Without users, it lacks sufficient support for automatic notifications. It also appeared to be stronger on back end features, with limited front-end abilities, and lacked support for uploading assets by users, so we felt that this system did not meet our requirements well enough to be considered in detail.

JBoss Portal is an open source web portal system based on open standards such as the Content Repository for Java Technology API (JSR-170). Its features include the portal and portal container, themes and layouts, user and group functionality, permissions management, content management system, and message boards. However, it appears to be primarily a set of Java libraries that need to be combined and built into an actual system. It also does not handle the web interface of a front end for the system. This makes it more complicated than other options, so it was not reviewed in detail.

Liferay Portal is an open source portal system designed to help provide a consolidated view of disparate applications. Its features include themes, sub-themes, personalization, CMS, application agnostic server, database agnostic, out of the box portlets, community based portlets, and administration of users, organizations, locations, and roles via a GUI. It appeared to be a generic content management system that did not seem to have support for asset reuse readily available. It would require a large amount of modifications and development effort to make it meet our requirements, so we did not review it in detail.

The Repository in a Box (RIB) software package was developed by the Innovative Computing Laboratory at the University of Tennessee. It is used to create web-based metadata repositories where metadata is considered by RIB to be information that describes reusable objects such as software. The repositories do not actually store any assets on the RIB system itself and since they do not host assets, we consider the resulting products to be catalogs, not repositories. RIB uses the Basic Interoperability Data Model (BIDM), which is an IEEE standard (1420.1), to improve interoperability of catalogs on the Internet. This provides a strong back-end for the system. However, when we examined this package, we found that it was missing many of the front-end features we require, such as the ability to provide reviews and ratings on assets. It was clear that many modifications would be necessary to adapt RIB for our purposes, so we did not review it in detail.

# 6.0 Review of Available Systems

Another option for creating a new RES is to augment an existing system. In this case, we would work together with the administrators of the existing system to determine what modifications would be necessary in order for it to meet all of our stated requirements to make those modifications. The RES would then be a new part of an existing system rather than a completely new stand-alone system. The systems we examined for this option include:

8) Global Change Master Directory (GCMD) (see reference 11 in Section 9)

Some other packages were examined, but not reviewed in detail. A brief description of each of the sites in the following list follows the detailed review of the sites listed above.

9) Ames Research Center Open Source Site (see reference 12 in Section 9)
10) Goddard Space Flight Center (BSFC) Open Source Site (see reference 13 in Section 9)

## 6.1 Global Change Master Directory (GCMD)

The GCMD is owned by NASA and is run by the Global Change Data Center within the Earth Sciences Directorate at the Goddard Space Flight Center. Its goal is “to enable users to locate and obtain access to Earth science data sets and services relevant to the global change and Earth science research.”

### 6.1.1 Installation

Since the GCMD is an existing system, installation would not be the same as for the software packages in the previous section. Here, installation would basically refer to the process by which we would add our planned RES into the existing GCMD system. It is not clear how easy or difficult this would be. According to the GCMD staff, creating a portal or view of a subset of materials stored in the GCMD is relatively easy and they have done so for other groups. However, not all of the reusable software assets we plan to have in the RES may be suitable for inclusion in the GCMD, so this option may be of limited value. Creating a separate instance of the GCMD system may be possible, but the staff indicated that it would be simpler for them to install and maintain it than for them to teach us how to do so.

### 6.1.2 Meeting Requirements

The following table indicates how well the GCMD meets our stated requirements. Explanatory notes follow, when applicable.

**Table 4 – GCMD Requirements Matching**

| Requirement Number and Title | Meets Requirement? |
|---|---|
| *1 – Users and User Information* | |
| *1.1 – Support for User Types* | |
| R1.1.1 – Support for Consumer User | YES |
| R1.1.2 – Support for Provider User | YES |
| R1.1.3 – Support for Administrator User | PARTIAL |
| R1.1.4 – Support for Content Manager User | PARTIAL |
| *1.2 – User Information Storage* | |
| R1.2.1 – Storage of Common User Information | NO |
| R1.2.2 – Storage of Provider Information | NO |
| *1.3 – User Interface* | |
| R1.3.1 – User Profile Management | NO |
| R1.3.2 – User Request Account Deletion | NO |
| *2 – Asset Storage and Management* | |
| *2.1 – Asset Information Storage* | |
| R2.1.1 – Storage of Asset Information | YES |
| R2.1.2 – Storage of Asset Resources | PARTIAL |
| R2.1.3 – Storage of Asset Versions | YES |
| R2.1.4 – Scanning of Asset Uploads | NO |
| *2.2 – Asset Discovery* | |
| R2.2.1 – Display Alphabetical Listing of Assets | NO |
| R2.2.2 – Provide Search for Assets | YES |
| R2.2.3 – Display Hierarchical Navigation of Assets | YES |
| *2.3 – Asset Management* | |
| R2.3.1 – Provider Registration of New Assets | YES |
| R2.3.2 – Provider Modification of Assets | YES |
| R2.3.3 – Provider Approval of Asset Modifications | NO |
| R2.3.4 – Provider Request for Asset Removal | YES * |

| Requirement Number and Title | Meets Requirement? |
|---|---|
| R2.3.5 – Provider Categorization of Assets | YES |
| *2.4 – Asset Feedback* | |
| R2.4.1 – Collection of Comments About Assets | NO |
| R2.4.2 – Collection of Quantitative Feedback | NO |
| R2.4.3 – User Registration of Asset Usage | NO |
| R2.4.4 – Feedback by Contacting Providers | YES |
| R2.4.5 – Display Feedback | NO |
| *2.5 – Asset Metrics and Reports* | |
| R2.5.1 – Collect Number of Downloads | NO |
| R2.5.2 – Collect Number of External Links Accessed | YES |
| R2.5.3 – Collect Number of Registered Users for Assets | NO |
| R2.5.4 –Summarize Ratings from Quantitative Feedback | NO |
| 2.6 – Asset Access Control | |
| R2.6.1 – Limit Access of Certain Users from Certain Assets | NO |
| *3 – Send and Manage Notifications* | |
| *3.1 – Send Notifications for Asset Events* | |
| R3.1.1 – Send Notification on Modification of Asset | YES |
| R3.1.2 – Send Notification on Submission of New Feedback | NO |
| *3.2 – Send Notifications for System Events* | |
| R3.2.1 – Send Administrative Notification for Asset Information | YES |
| R3.2.2 – Send Administrative Notification for System Information | YES |
| *3.2 – Notification Management* | |
| R3.2.1 – User Addition of Notifications for Assets | YES |
| R3.2.2 – User Removal of Notifications | YES |
| *4 – System Operations* | |
| *4.1 – System Feedback* | |
| R4.1.1 – Collection of System Problems | YES |
| R4.1.2 – Collection of Suggestions | YES |
| R4.1.3 – Feedback by Contacting Administrators | YES |
| *4.2 – System Policies Compliance, Security, and Privacy* | |
| R4.2.1 – Verification of Provider Information | NO |
| R4.2.2 –Verification of Provider through Secondary Method or Contact | NO |
| R4.2.3 – Security of Sensitive Transmitted Information | NO |
| R4.2.4 – Security of Stored Information | NO |
| R4.2.5 – Deletion of Users for Policy Enforcement | NO |
| R4.2.6 – Protection of Private Information | YES |
| R4.2.7 – Compliance with other Technical, Accessibility, and Security Requirements | YES |
| R4.2.8 – Policies Availability to Users | YES |
| *4.3 – Repository and Catalog* | |
| R4.3.1 – Function as a Repository | PARTIAL |
| R4.3.2 – Function as a Catalog | YES |

| **Requirement Number and Title** | **Meets Requirement?** |
|---|---|
| R4.3.3 – Selection of System Behavior by Provider | NO |
| R4.3.4 – Enforcement of Asset Storage Limit | NO |
| *4.4 – Asset Cleanup* | |
| R4.4.1 – Asset Deprecation by Content Managers | YES |
| R4.4.2 – Asset Removal by Administrators | YES |
| *4.5 – Data Integrity* | |
| R4.5.1 –Verification of Data by Providers | NO |

Requirement 1.1.3, Support for Administrator User, is partially met because user accounts are not available, so Administrators cannot approve new accounts. All of the other parts of this requirement (e.g., managing assets and approving submissions, modifications, and deletions) are met.

Requirement 1.1.4, Support for Content Manager User, is partially met because there is no specific user role with these duties, but the role of content management is performed by Administrators.

Requirement 2.1.2, Storage of Asset Resources, is partially met because assets may only be stored remotely and linked to by the system. Assets may not be uploaded and stored/hosted on the system directly.

Requirement 2.3.4, Provider Request for Asset Removal, is met through Requirement 2.3.2, Provider Modification of Assets, since Providers can remove access to their assets by modifying the information available for it. Also, Requirement 4.1.3, Feedback by Contacting Administrators, allows Providers to contact administrators in order to fully remove the asset.

Requirement 4.3.1, Function as a Repository, is partially met because the system has the capability of storing very small data sets, typically only in the case where the data sets are in danger of being lost. However, it cannot function as a general repository for all users and assets, and therefore the following requirements that depend on having repository functionality are not met:

- R2.1.4 – Scanning of Asset Uploads
- R2.5.1 – Collect Number of Downloads
- R4.3.3 – Selection of System Behavior by Provider
- R4.3.4 – Enforcement of Asset Storage Limit
- R4.5.1 – Verification of Data by Providers

Since the system does not have a feature that allows registration of user accounts, the following requirements, all dependent on this ability, are not met:

- R1.2.1 – Storage of User Information
- R1.2.2 – Storage of Provider Information

- R1.3.1 – User Profile Management
- R1.3.2 – User Request Account Deletion
- R4.2.1 – Verification of Provider Information
- R4.2.2 – Verification of Provider through Secondary Method or Contact
- R4.2.5 – Deletion of Users for Policy Enforcement

#### 6.1.3 Gap Analysis

The most obvious gaps would be the limited repository functionality and the lack of user accounts, which means many of our requirements are not met. As the GCMD recommends that they administer a separate instance of the system, it is not clear exactly how much effort it would take to modify the GCMD system to meet our requirements. However, the indication is that many modifications would be necessary, so the effort would be high.

#### 6.1.4 Maintenance and Support

Since the GCMD recommended that they maintain the system, because that would be easier than teaching us how to do it, the maintenance and support for the Software Reuse Working Group would be minimal – the GCMD staff would take care of it. However, the large number of changes that would need to be made for an instance to meet our requirements implies that a large amount of maintenance and support would be required for the system, since it would be a non-standard version of the system.

#### 6.1.5 Summary

Significant effort would be required to modify the GCMD system to meet the identified requirements. The GCMD is an existing system under the control of another project and adding capabilities to support a reuse enablement system is currently not a goal of that project. If a separate instance of the system were to be created for the reuse enablement system, estimates indicate that it would take about the same amount of development effort as Savane to modify the GCMD instance to meet our requirements. However, because of the GCMD's large user base and the close link between data and software, it is important that any new reuse enablement system provide the GCMD with a data feed of relevant content (Software Reuse WG, 2005, RES Trade Study). Out of our 54 requirements, the GCMD meets 26, partially meets 4, and does not meet 24.

### 6.2 Other Systems Inspected

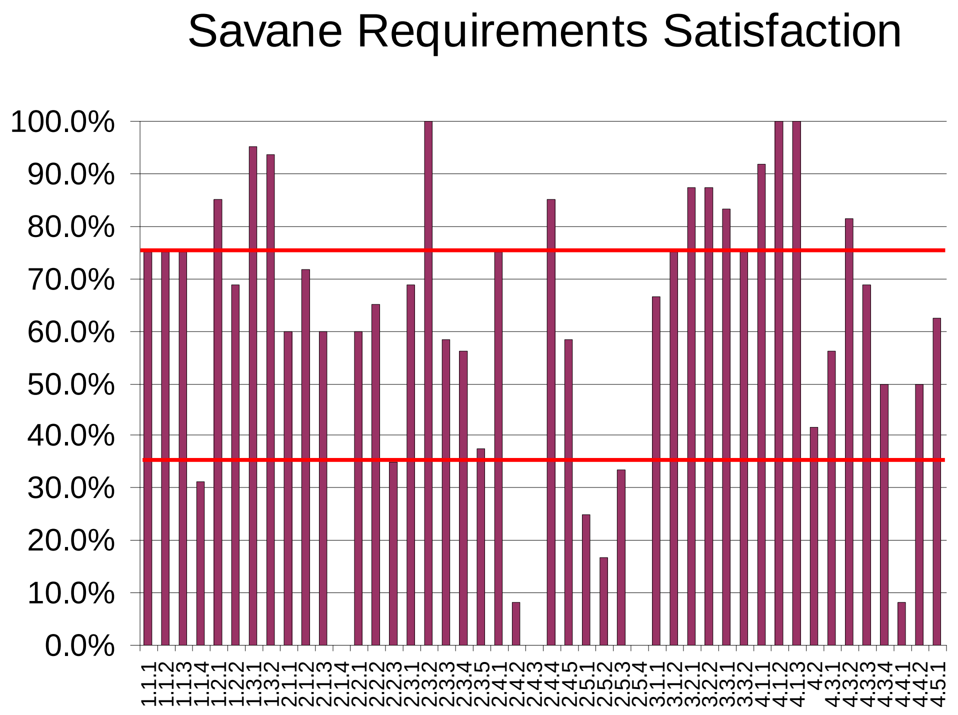
There are other systems that provide software asset catalogs and/or repositories in addition to the ones reviewed here. Two of these are the open source sites hosted by the NASA Ames Research Center and the NASA Goddard Space Flight Center. There are two basic reasons these sites were not reviewed in detail: the limited scope of their content and the limited functionality they provide.

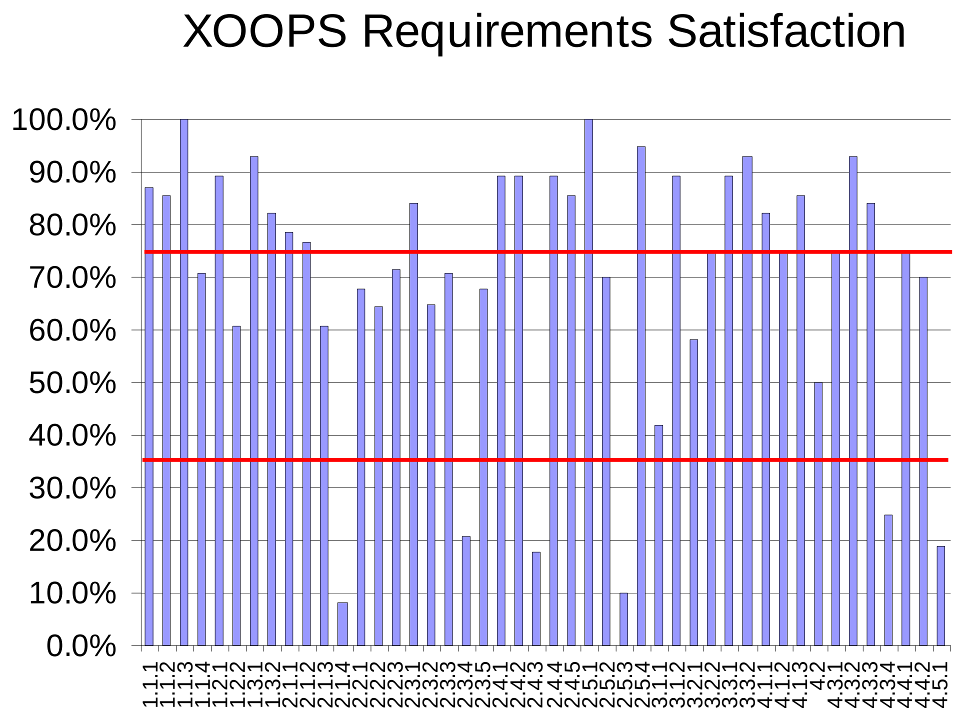
As discussed further in the Reuse Enablement System (RES) Trade Study document, these sites only distribute NASA-produced open source software, which places a strong limitation on what is available. This prevents them from fully servicing the needs of the community of Earth science software developers. In addition, they also have the common problem of primarily listing finished products, which could be difficult for software developers to reuse when creating new assets, rather than smaller software components, which developers desire more for reuse purposes.

Both sites list assets as links on a web page. The sites have grown somewhat since the trade study was performed in fall 2005 and provide more assets now, but neither site has added an underlying catalog/repository system. While this works well for their purposes, it would not meet the community's needs, where many other features are desired that could not easily be offered without a real catalog/repository system behind the site. For these reasons, these sites are not suitable for meeting the requirements of the RES, so we did not review them in detail.

## 7.0 Review Summary

As part of our evaluation process, members of the Working Group tested prototypes of the Savane and XOOPS systems, our primary candidates for building the RES, at the 5th ESDS Working Group Meeting, held November 14–16, 2006. An evaluation form was distributed, and asked for a rating from 1 to 5 of how well each system met the requirements we had defined. We received a total of seven responses, but not every person answered every question. The 1–5 ratings were converted to percentages, with 1 being 0% and 5 being 100%, in order to make an estimate of how satisfied the respondents were with the way the systems met our requirements. All results were averaged together and graphed to provide a visual representation of how well the systems satisfied our requirements. These graphs are presented below, with totals for the number of requirements satisfied (75% or higher), not satisfied (35% or lower), and partially satisfied (between 35% and 75%). We also note that Savane and GForge are and both collaborative development environments, so they share similar features and requirements. Savane was estimated to be more flexible and easier to maintain than GForge, so we focused our efforts on Savane.





- 18 Requirements Satisfied (75%-100%)
- 18 Requirements Partially Satisfied (35%-75%)
- 10 Requirements Not Satisfied (0%-35%)

- 26 Requirements Satisfied (75%-100%)
- 14 Requirements Partially Satisfied (35%-75%)
- 6 Requirements Not Satisfied (0%-35%)

It is important to note that (lack of) satisfaction with the requirements does not necessarily correspond to (not) meeting the requirements. For example, it is possible for the system to meet a requirement technically, but respondents evaluate it as partially or not satisfied because of the implementation. The evaluations of Savane and XOOPS in Sections 5.2 and 5.3 are based on the technical ability of the system to meet the requirements. This does not consider how well the system meets the requirements, only if it does or not. Requirements can be met even if the implementation is less than ideal. The evaluations performed by the members of the Working Group were more subjective in nature, providing ratings of how satisfied the respondents were with the way the system met the requirements. This allows requirements that are technically met in some way to be given lower ratings if they do not meet the requirements in the most satisfactory way. Since the two evaluations had some differences in their perspective, some differences in the results are expected. However, given those differences, the low number statistics in the Working Group evaluations, the generally fewer responses for Savane than XOOPS (five or less compared to seven or less), the lack of a one-to-one matching between evaluation questions and requirements, and the variation in number of responses per question, the general agreement between the Working Group evaluations and the ones in Sections 5.2 and 5.3 are good. Requirements rated highly by the working group get a "yes" while low-rated requirements get a "no". The main difference is in the area of partially met/satisfied requirements. The dividing lines between satisfied, partially satisfied, and not satisfied in the Working Group evaluations are somewhat arbitrary, and shifting them could allow a closer match with the evaluations of Sections 5.2 and 5.3. The Working Group satisfaction evaluations provide general support for the technical evaluations, so the technical evaluations are used as the basis of the gap analysis, development effort estimates, and decision/conclusion.

# 8.0 Decision and Conclusion

As described in Sections 5 and 6, summarized in Table 5 below, XOOPS meets more and fails fewer of our requirements than Savane. XOOPS met 40 requirements compared to Savane's 24, and only failed 9 requirements compared to Savane's 20. It also has fewer requirements that are partially met (5 compared to 10). Since XOOPS does a better job at meeting our requirements overall, the amount of effort required to modify the system to meet our requirements is less than that for Savane. In addition, XOOPS uses modules to provide functionality for the system, and each module is a self-contained component. This makes it easier to modify XOOPS than Savane since our modifications will be restricted to particular new or existing modules. When the base XOOPS system is upgraded, there is little chance that it will affect our modifications. Since our modifications will be isolated from the base system, maintaining our changes will be simpler. Therefore, we have determined that XOOPS is the most suitable choice for developing a Reuse Enablement System (RES) from an existing software package.

We also examined the Global Change Master Directory (GCMD), an existing system, for its possible use as an RES for the community of Earth science software developers. In our evaluation, it was roughly comparable to Savane in terms of requirements met, partially met, and unmet, and therefore it is estimated that the development effort for modifying the GCMD to meet these requirements would be similar to the effort to implement Savane. However, since the GCMD is an existing system under the control of another project, such modifications would need to go through the existing team for updates, support, and maintenance. In addition, adding capabilities to support a reuse enablement system is currently not a goal of the GCMD project. Even if a separate instance of the system were to be created for our use, the GCMD staff has indicated that it would be preferable for them to install and maintain it than for them to teach us how to do so. Therefore, it seems preferable to create the RES from an existing software package rather than to modify an existing system like the GCMD to meet our needs and requirements.

**Table 5 – Summary of Results**

| Approach Studied | # Requirements Met | # Requirements Not Met | # Requirements Partially Met | Development Effort Estimate [staff-months] |
|---|---|---|---|---|
| XOOPS | 40 | 9 | 5 | 8.12 |
| Savane | 24 | 20 | 10 | 34.01 |
| GCMD | 26 | 24 | 4 | N/A |
| GForge | 20 | 26 | 8 | N/A |

Comparing the level of effort to create the RES using XOOPS and Savane, XOOPS requires the least amount of development, which indicates that XOOPS, an open source content management system, should be used to create a prototype RES for internal NASA use.

## 8.1 Next Step

The long-term objective remains the establishment of a fully functional Reuse Enablement System that will reduce the cost and development time for new Earth science and possibly space science systems to allow NASA to better support scientific discovery and understanding. We have accomplished important steps in defining the requirements for such a system and in identifying XOOPS as a cost effective and compliant basis for the system. From here, plans for the design and implementation of the system will be developed. Following that, a plan will be created to establish and evaluate a working prototype system for internal NASA use using XOOPS. Based on the evaluation of the working prototype system, a system for public use could be developed in the future.

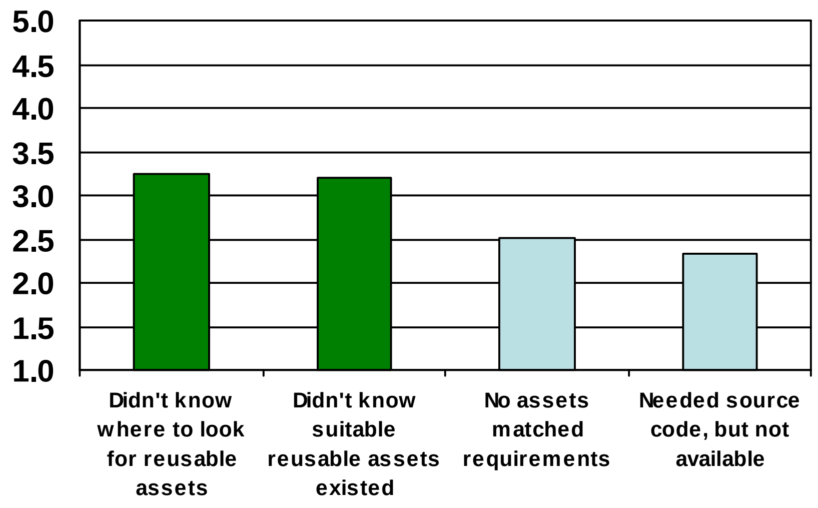
# 9.0 References

[1] Software Reuse portal web site – http://www.esdswg.org/softwarereuse
[2] Software Reuse collaboration site – http://www.sciencedatasystems.org/reuse/default.aspx
[3] GForge web site – http://gforge.org/
[4] Savane web site – https://gna.org/projects/savane
[5] XOOPS web site – http://www.xoops.org/
[6] Fedora Digital Repository System – http://www.fedora.info/
[7] JBoss Portal – http://www.jboss.org/products/jbossportal
[8] Liferay Portal – http://www.liferay.com/web/guest/products/portal
[9] Repository in a Box – http://icl.cs.utk.edu/rib/
[10] SourceMotel – https://sourcemotel.gsfc.nasa.gov/
[11] Global Change Master Directory (GCMD) – http://gcmd.nasa.gov/
[12] Ames Research Center Open Source site – http://opensource.arc.nasa.gov/
[13] Goddard Space Flight Center (GSFC) Open Source site – http://opensource.gsfc.nasa.gov/

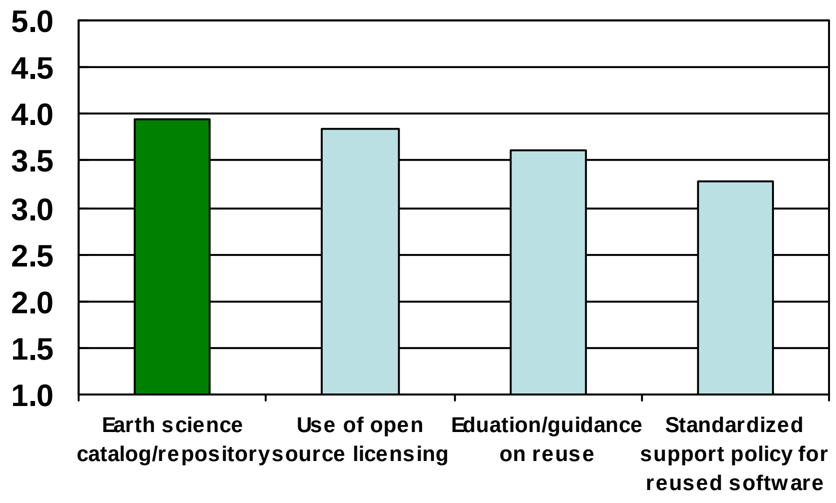
## Appendix A – Software Reuse Questionnaire

The majority of the survey consisted of multiple choice questions where each listed option was ranked from 1 (not important at all) to 5 (very important). The following charts show the weighted average results for the top few responses to two of the questions.

Question 7 – How important were the following factors in preventing you from reusing software development artifacts developed outside your group?



Question 47 – In your opinion, how important would the following factors be in helping increase the level of reuse within the Earth science community?



# Appendix B – Enabling Systems Recommendation

The Software Reuse Working Group previously submitted a recommendation for a Reuse Enablement System to NASA HQ. This appendix contains the content of that recommendation and HQ's response to it.

- NASA should establish a system to facilitate the cataloging and distribution of reusable assets for the Earth science community
- NASA should establish an effective mechanism for dissemination of reusable assets within the Earth science community
- NASA should evaluate the technology options for the provision of a reuse enablement system including:
    - commercial reuse catalogs/repositories
    - open source reuse catalogs/repositories
    - use of existing publicly available catalogs/repositories
    - custom build of a community-specific catalog
- Based on the conclusions of the technology evaluation, NASA should implement a reuse enablement system
- NASA should develop guidelines and standards for the management and operation of a reuse enablement system

**Impact for the Working Group**

- The reuse working group will evaluate the technology options for the provision of a reuse enablement system
- The reuse working group will develop guidelines and standards for the management and operation of a reuse enablement system
- The reuse working group will develop a proposal for the implementation of a reuse enablement system based on the conclusions of the technology evaluation
- One additional FTE will be required for the balance of '05 fiscal year

**Desired Decision**

- HQ agreement to proceed with the evaluation of technology options and to provide funding for the evaluation
- HQ agreement in principle to the establishment of a reuse catalog subject to the findings of the evaluation

**Headquarters' Response**

- HQ thinks such a recommendation is premature and needs to await the results of a trade study concerning the establishment of a reuse catalog

# Appendix C – General Requirements

For several months in 2004, the Software Reuse Working Group collaborated to define a set of requirements for a software Reuse Enablement System serving the Earth science community. This appendix contains a list of those requirements as previously submitted to NASA HQ.

**General Requirements**

- The system will facilitate the distribution and reuse of software development artifacts across the Earth and space science communities
- The reusable artifacts supported by the system will include software components and other digital artifacts used in the software development process
- The system will run on industry standard hardware and operating system
- The system will support remote access through standard Internet browsers
- The system will support the automated collection of system and asset usage metrics
- The system will provide error handling for all capabilities
- The system shall be flexible to support changes in NASA policy and strategy

**Search Requirements**

- The system will allow users to browse and look at system content without registering
- The system will allow users to discover (search for and find) assets of interest using multiple search mechanisms (e.g., keyword search or category search)
- The system will allow search results to be ordered in a number of ways (e.g., by category or rating)

**User Registration**

- The system will allow new users to register with the system and the user role defined by the registration will determine the user's access authority within the system
- Each user registration will require the approval of a system administrator
- The system will allow a user to update their user profile
- The system will allow registered users to provide system feedback
- The system will allow registered users to subscribe to system or asset events including events such as new versions, updates, and comments supplied by other users

**Asset Usage**

- The system will allow a Consumer to acquire an asset from the system repository
- The system will allow a Consumer to register usage of an asset, indicating active usage of the asset (this is different from downloading the asset)
- The system will provide a user forum for discussion and comments on assets
- The system will allow a Consumer to provide a rating and feedback on his/her experience with a particular asset
- The system will allow posting of requests for reusable assets that currently are not in the system

**Asset Submission**

- The system will allow a Provider to submit a new asset profile to the system
- The Provider may optionally upload artifacts associated with the asset
- Each asset submission will require the approval of a Content Manager before it can be accessed by other system users
- The system will allow a Provider to update the information about an asset and change the artifacts associated with the asset
- The system will allow Providers to subscribe to asset events including such events as comments and new requests pertaining to their contribution

**Content Management**

- The system will allow users to review feedback on assets and allow the Content Manager to remove feedback on assets (e.g., to make sure comments are on topic)
- The system will allow the Content Manager to review and approve asset submissions prior to them being made available to the community
- The system will allow the Content Manager to review the assets and remove those which are no longer relevant; this includes those that have poor reviews and/or no users
- The system will allow the Content Manager to review unsuccessful searches to capture consumer demand for assets that are not registered

**System Administration**

- The system will allow Administrators to monitor the general operating state of the system and perform designated routine tests to determine that the system is functioning properly
- The system will allow Administrators to manage user accounts and passwords
- The system will allow Administrators to monitor user feedback and use it to determine evolutionary needs of the system and other users
- The system will allow Administrators to generate reports including metrics
- The system will send notifications to subscribed users of system issues or events

## Appendix D – COCOMO 81 Description

The COCOMO (Constructive Cost Modeling) model developed by Barry Boehm provides a method for estimating the cost, effort, and schedule involved in software development activities. For our purposes here, we have made the estimate using COCOMO 81, the original version, which provides a relatively simple way to estimate effort. We used the information available from the University of Calgary's Practical Software Engineering site (http://ksi.cpsc.ucalgary.ca/courses/451-96/mildred/451/CostEffort.html#RTFToC9) to guide our estimates. Additional information can be found at, for example, the Center for Software Engineering's page at http://sunset.usc.edu/research/COCOMOII/index.html.

The basic concept of the COCOMO 81 model is that the development effort can be expressed as $E=a*(size)^b$, with E in staff-months and size in KLOC. The factors a and b are constants that change according to the estimate required. Projects are categorized as organic, semi-detached, and embedded, primarily by their size.

| **Project Type** | **Characteristics** | | | |
|---|---|---|---|---|
| | *Size* | *Innovation* | *Deadline/ Constraints* | *Development Environment* |
| Organic | Smallish | Little | Not tight | Stable |
| Semi-detached | Medium | Medium | Medium | Medium |
| Embedded | Large | Greater | Tight | Complex hardware / customer interfaces |

The basic model, which we used, only uses source size to determine the values of the constants a and b, as given in the following table.

| | **Organic** | **Semi-detached** | **Embedded** |
|---|---|---|---|
| a | 2.4 | 3.0 | 3.6 |
| b | 1.05 | 1.12 | 1.20 |

Therefore, the effort (E) is given by the following equations, where S is the source size.

| **Mode** | **Effort Formula** |
|---|---|
| Organic | $E = 2.4 * (S^{1.05})$ |
| Semi-detached | $E = 3.0 * (S^{1.12})$ |
| Embedded | $E = 3.6 * (S^{1.20})$ |

There is also an intermediate model, which uses 15 additional cost drivers as well as size, but for our purposes, the simple model provides a useful estimate of the level of effort required for our gap analysis.

## Appendix E – Savane Gap Analysis

Below is the list of requirements not met by Savane, covered in Section 5.2, and an analysis for the level of effort to meet them.

The components or subsystems considered for Savane include the database, the PHP frontend, the Perl backend, Savane configuration, and other external software such as Mailman, CVS, Subversion, etc.

As a guideline to help determine the complexity of the changes, and the maintenance they require, we scored each requirement's changes on a scale of 0 to 10. We used the following calculations as a basis for this score:

**Complexity**

Base: 2/10

- +1 PHP frontend
- +2 Database
- +1 Perl backend
- +2 Configuration
- +0 Layout/theme change
- +2 per external software

### R1.1.3 – Support for Administrator User

**Evaluation**

The administrator has no interface to approve/disapprove changes to the project's entry in the system. Once a project has been approved, the project managers may change the details of the project without further approval.

**Subsystems Requiring Modification**

- **PHP frontend:** accept changes without updating the database, notify the administrator(s) of a change, provide UI for administrators to approve changes, change project deletion function in UI to require administrator approval
- **Database:** add tables/fields for pending changes

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 50 |
| PHP | 500 |
| ***Total*** | **550** |

**R1.1.4 – Support for Content Manager User**

**Evaluation**

Project managers/owners can act as content managers for their respective project, but the system does not support the concept of content managers for groups of projects. This requirement also depends on unmet requirement 2.3.3 (Provider Approval of Asset Modifications).

**Subsystems Requiring Modification**

- **PHP frontend:** create UI for approving or rejecting pending changes to assets, create UI to set certain users as content managers, create UI to define groups of content management: both which users are content managers for a group, and which projects belong to that group.
- **Database:** add tables/fields for content manager role, project groups that can be managed, and which content managers can manage which groups.

**Complexity** 9/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| SQL | 100 |
| PHP | 1500 |
| ***Total*** | **1600** |

**R1.2.2 – Storage of Provider Information**

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** add fields that are required upon submission of a new project, auto fill fields if already approved for another project to verify, protect user input
- **Database:** add tables and/or fields, queries for data management

**Complexity** 5/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| SQL | 50 |
| PHP | 100 |
| ***Total*** | **150** |

## R2.1.1 – Storage of Asset Information

### Evaluation

The system does not store keyword information, nor allows for multiple categories.

### Subsystems Requiring Modification

- **PHP frontend:** require extra fields for new projects, display new project data in project summary, allow for changing these fields by project owner, protect user input, validate tags/keywords
- **Database:** add tables and data for keywords and categories, with categories in heirarchy
- **Perl backend:** update database with new project details, when project gets created

**Complexity** 6/10

### Estimate Lines of Code

| ***Language*** | ***Lines of Code*** |
|---|---|
| SQL | 50 |
| Perl | 100 |
| PHP | 500 |
| ***Total*** | **650** |

## R2.1.4 – Scanning of Asset Uploads

### Evaluation

Not supported.

### Subsystems Requiring Modification

- **PHP frontend:** on upload of a new file, process the file through virus checking, and quarantine any "bad" file
- **Perl backend:** create extra storage for bad files for each project
- **Configuration:** add configuration about the storage of bad files
- **External software:** add virus scanner software
- **External software:** disable version control system, or provide a warning that version control system is not virus checked

**Complexity** 10/10

### Estimate Lines of Code

| ***Language*** | ***Lines of Code*** |
|---|---|
| SQL | 50 |
| Perl | 100 |
| PHP | 100 |
| Configuration | 10 |
| Virus Scan/Cron | 5 |
| ***Total*** | **275** |

**R2.2.1 – Display Alphabetical Listing of Assets**

**Evaluation**

The system does not provide a full alphabetical list of all assets.

**Subsystems Requiring Modification**

- **PHP frontend:** create a link that searches for *

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 5 |
| ***Total*** | **5** |

**R2.2.3 – Display Hierarchical Navigation of Assets**

**Evaluation**

The system does not provide more than a one-level categorization of items. The hierarchical navigation UI depends on unmet requirement 2.3.5 (Provider Categorization of Assets).

**Subsystems Requiring Modification**

- **PHP frontend:** update display/navigation, provide UI for modification of categories
- **Database:** add tables/fields for storing categories
- **Perl backend:** update database with new asset information when project gets created

**Complexity** 6/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| SQL | 50 |
| Perl | 100 |
| PHP | 500 |
| ***Total*** | **650** |

**R2.3.3 – Provider Approval of Asset Modifications**

**Evaluation**

The system has no mechanism to allow managers to approve/disapprove changes to an asset. Providers can be approved to manage an asset, but individual changes are not provided for review.

**Subsystems Requiring Modification**

- **PHP frontend:** create UI for uploading assets for pending submission, notify manager of pending changes, provide restricted access to files for manager to view
- **Database:** add tables/fields for pending changes
- **Perl:** maintain database, provide notifications, move approved files after approval
- **Configuration:** add pending files directory and configuration
- **External software:** restrict ssh upload of files, only allow upload through UI
- **External software:** optionally restrict or remove usage of version control system

**Complexity** 10/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 1000 |
| SQL | 50 |
| Perl | 50 |
| Configuration | 50 |
| ***Total*** | **1150** |

## R2.3.5 – Provider Categorization of Assets

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** provide UI for submission of category modification, notification to administrator, UI for administrator approval/rejection
- **Database:** add tables/fields for storing pending category change

**Complexity** 5/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 500 |
| SQL | 10 |
| ***Total*** | **510** |

## R2.4.1 – Collection of Comments About Assets

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** UI to submit comment
- **Database:** create fields/tables to store comments

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 50 |
| PHP | 500 |
| ***Total*** | **550** |

### R2.4.2 – Collection of Quantitative Feedback

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** user UI to submit or change rating feedback
- **Database:** create fields/tables to collect ratings

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 10 |
| PHP | 500 |
| ***Total*** | **510** |

### R2.4.3 – User Registration of Asset Usage

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** UI to register/unregister for usage of software
- **Database:** create fields/tables to keep track of registration

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 10 |
| PHP | 100 |
| ***Total*** | **110** |

### R2.4.4 – Feedback by Contacting Providers

**Evaluation**

The system allows any registered users to contact each other through the web interface, but does not allow anyone to turn this feature off.

**Subsystems Requiring Modification**

- **PHP frontend:** UI to allow provider to turn off contact feature
- **Database:** create fields/tables to keep track of contact setting

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 10 |
| PHP | 50 |
| ***Total*** | **60** |

### R2.5.1 – Collect Number of Downloads

**Evaluation**

Not supported

**Subsystems Requiring Modification**

- **PHP frontend:** create a link to a separate page for each download (refreshes immediately to download)
- **Database:** create fields/tables to keep track of how many times the download page was accessed

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 10 |
| PHP | 500 |
| ***Total*** | **510** |

## R2.5.2 – Collect Number of External Links Accessed

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** create a link to a separate exit page (refreshes immediately to link)
- **Database:** create fields/tables to keep track of how many times the link was accessed

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| SQL | 10 |
| PHP | 500 |
| ***Total*** | **510** |

## R2.5.3 – Collect Number of Registered Users for Assets

**Evaluation**

Not supported. This depends on unmet requirement 2.4.3 (User Registration of Asset Usage).

**Subsystems Requiring Modification**

- **PHP frontend:** report that shows members stats about their assets.

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 500 |
| ***Total*** | **500** |

### R2.5.4 – Summarize Ratings from Quantitative Feedback

**Evaluation**

Not supported. This depends on unmet requirement 2.4.2 (Collection of Quantitative Feedback).

**Subsystems Requiring Modification**

- **PHP frontend:** report that shows members stats about their assets' feedback

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 500 |
| ***Total*** | **500** |

### R2.6.1 – Limit Access of Certain Users from Certain Assets

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** manager UI to restrict categories or individual assets to groups of users or individual registered users, block asset pages for restricted users
- **Database:** create fields/tables to keep track of which users or groups can access which assets or categories

**Complexity** 5/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 500 |
| SQL | 100 |
| ***Total*** | **600** |

### R3.1.1 – Send Notification on Modification of Asset

**Evaluation**

Not supported. This depends on unmet requirements 2.3.3 (Provider Approval of Asset Modifications) and 3.3.1 (User Addition of Notifications for Assets).

**Subsystems Requiring Modification**

- **PHP frontend:** when an administrator or manager approves changes, send an e-mail to anyone who is registered for notifications

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

### R3.1.2 – Send Notification on Submission of New Feedback

**Evaluation**

Not supported. This depends on unmet requirements 2.4.1 (Collect Feedback).

**Subsystems Requiring Modification**

- **PHP frontend:** UI for providers to indicate they want to be notified on new feedback, e-mail a provider whenever new feedback is left.

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

### R3.2.1 – Send Administrative Notification for Asset Information

**Evaluation**

Notifications can be posted to the asset, but there is no mechanism to send them out. This depends on unmet requirements 3.3.1 (User Addition of Notifications for Assets).

**Subsystems Requiring Modification**

- **PHP frontend:** send e-mail notification to all users requesting notifications whenever something new is posted to the asset.

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

## R3.2.2 – Send Administrative Notification for System Information

**Evaluation**

Notifications can be posted about the system, but there is no mechanism to send them out. This depends on unmet requirements 3.3.1 (User Addition of Notifications for Assets).

**Subsystems Requiring Modification**

- **PHP frontend:** send e-mail notification to all users requesting notifications whenever something new is posted to the asset.

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

## R3.3.1 – User Addition of Notifications for Assets

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **PHP frontend:** UI options for users to receive notifications about: the system, categories, or individual assets, offer notification option when registering asset usage
- **Database:** create tables/fields to keep track of different notifications for each user

**Complexity** 5/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 500 |
| SQL | 100 |
| ***Total*** | **600** |

**R3.3.2 – User Removal of Notifications**

**Evaluation**

Not supported. This depends on unmet requirement 3.3.1 (User Addition of Notifications for Assets).

**Subsystems Requiring Modification**

- **PHP frontend:** UI with options to stop receiving notifications

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

**R4.2.1 – Verification of Provider Information**

**Evaluation**

Not supported. This depends on unmet requirement 1.2.2 (Storage of Provider Information).

**Subsystems Requiring Modification**

- **PHP frontend:** user interface to approve/reject a provider
- **Perl backend:** update database with new user information upon approval

**Complexity** 6/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 500 |
| Perl | 100 |
| ***Total*** | **600** |

**R4.2.2 – Verification of Provider through Secondary Method or Contact**

**Evaluation**

Not supported. This depends on unmet requirement 4.2.1 (Verification of Provider Information).

**Subsystems Requiring Modification**

- **PHP frontend:** required fields for contact information
- **Database:** add required fields for contact information

**Complexity** 5/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 100 |
| SQL | 10 |
| ***Total*** | **110** |

**R4.2.3 – Compliance with Other Technical, Accessibility, and Security Requirements**

**Evaluation**

The system does not follow all of the policies for NASA web sites, such as the standard NASA header and footer information.

**Subsystems Requiring Modification**

- **PHP frontend:** modify files for display
- **Layout:** add header and footer information.

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 50 |
| HTML | 100 |
| ***Total*** | **150** |

**R4.2.8 – Policies Availability to Users**

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- **Layout:** add documentation to site using tools

**Complexity** 2/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| HTML | 100 |
| ***Total*** | **100** |

### R4.3.4 – Enforcement of Asset Storage Limit

**Evaluation**

Not supported.

**Subsystems Requiring Modification**

- Configuration: create filesystem quotas for users and/or groups.

**Complexity** 4/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| Configuration | 10 |
| ***Total*** | **10** |

### R4.4.1 – Asset Deprecation by Content Managers

**Evaluation**

Not supported. This depends on unmet requirement 1.1.4 (Content Manager).

**Subsystems Requiring Modification**

- **PHP frontend:** user interface for toggling deprecation flag
- **Database:** add field for deprecation flag

**Complexity** 5/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 100 |
| SQL | 5 |
| ***Total*** | **105** |

# Appendix F – XOOPS Gap Analysis

Below is the list of requirements not met by XOOPS, covered in Section 5.3, and an analysis for the level of effort to meet them.

The components or subsystems considered for XOOPS include the database, the PHP engine, XOOPS configuration, and other external software such as Mailman, CVS, Subversion, etc.

As a guideline to help determine the complexity of the changes, and the maintenance they require, we scored each requirement's changes on a scale of 0 to 10. We used the following calculations as a basis for this score:

**Complexity**

Base: 0/10

- +1 PHP engine
- +2 Database
- +1 Configuration
- +1 Layout/theme change
- +2 per external software

## R2.1.4 – Scanning of Asset Uploads

**Evaluation**

Not supported. Currently, the system can accept uploaded files (via a download module), but has no mechanism to automatically run virus scanning software. We need a mechanism to run a virus scan and handle the uploaded file.

**Subsystems Requiring Modification**

- **PHP frontend:** on upload of a new file, process the file through virus checking tool and quarantine any “bad” file.
- **Configuration:** add configuration about the storage of bad files.
- **External software:** add virus scanner software

**Complexity** 4/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 300 |
| Configuration | 10 |
| Virus Scan/Cron | 5 |
| ***Total*** | **315** |

**R2.2.1 – Display Alphabetical Listing of Assets**

**Evaluation**

Not supported. This feature is not provided by the default download module, but is provided by at least one download module available at the official XOOPS site's module repository. This module or one with similar functionality would have to be installed in order to meet the requirement. Module installation is a simple and easy process, and this has been done on the prototype test site already. The level of effort needed to fulfill this requirement is therefore trivial.

**Subsystems Requiring Modification**

- **External software:** module for alphabetical listing.

**Complexity** 2/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| Configuration | 10 |
| ***Total*** | **10** |

**R2.3.3 – Provider Approval of Asset Modifications**

**Evaluation**

The system does provide a mechanism for approving modifications, but Providers are not the ones to do this by default, Administrators are. However, Administrators can work together with Providers on the approval process.

Requirement 2.3.3, Provider Approval of Asset Modifications, is partially met because the Provider is not by default the user who approves changes. The Administrators and/or Content Managers will have this role, but can work together with Providers to meet the requirement. This requirement would be met through this collaborative effort to approve modifications to assets, and would not require additional coding in this case. Modifying the system to redirect changes to the Providers would be a more difficult task.

**Subsystems Requiring Modification**

- **PHP frontend:** create UI for uploading assets for pending submission and/or notify providers of changes that match their asset, provide providers with some capability to view changes
- **Database:** add tables/fields for pending changes and to associate providers with assets.

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 1000 |
| SQL | 50 |
| ***Total*** | **1050** |

**R2.4.3 – User Registration of Asset Usage**

**Evaluation**

There is no way for users to register their active usage of an asset. Another module that provides this feature has not been located, so we would have to code this feature ourselves. The difficulty of this task would depend on how advanced we wanted to make the feature and how we decided to incorporate it into the existing XOOPS system. One of the simpler solutions is to provide a web form where users can provide some form of contact information and the name of the asset the wish to register, and submitting the form sends the information to the Administrator(s) of the system. More advanced solutions would be to write a new module providing this feature or modifying the code of the download module to provide a similar feature.

**Subsystems Requiring Modification**

- **PHP frontend:** UI to register/unregister usage of software
- **Database:** create fields/tables to keep track of registration

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 500 |
| SQL | 10 |
| ***Total*** | **510** |

**R2.4.4 – Feedback by Contacting Providers**

**Evaluation**

The default downloads module only allows a web site link for additional contact information and does not identify the submitter. For web links and assets stored remotely, we assume the site hosting the asset would contain the required contact information. This requirement can be more fully met by installing a different downloads module. At least one downloads module available at the official XOOPS site's module repository provides more contact information for providers with the submitted downloads in addition to a web site address.

**Subsystems Requiring Modification**

- **PHP frontend:** store and associate provider information with download, UI to allow provider to turn off contact feature
- **Database:** create fields/tables to keep track of contact setting

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 200 |
| SQL | 10 |
| ***Total*** | **210** |

### R2.5.3 – Collect Number of Registered Users for Assets

**Evaluation**

Not supported. This requirement is not met because Requirement 2.4.3, User Registration of Asset Usage, is not met. In the process of adding the feature to register asset usage, a method for collecting metrics on the number of users who have registered usage of a particular asset would be created.

**Subsystems Requiring Modification**

- **PHP frontend:** report that shows members stats about their assets.

**Complexity** 1/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 500 |
| ***Total*** | **500** |

### R4.2.2 – Verification of Provider through Secondary Method or Contact

**Evaluation**

Not supported. There is no place for users to put such secondary contact information on their registration. As for Requirement 1.2.2, such information could be added to the user profile after registering. This is one option, but would require all users who wish to be Providers to register as Consumers first, and then provide the necessary information required to process a change to the Provider role. Another solution would be to modify the XOOPS code to request the necessary information during the registration process.

**Subsystems Requiring Modification**

- **PHP frontend:** required fields for contact information
- **Database:** add required fields for contact information

**Complexity** 3/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 200 |
| SQL | 10 |
| ***Total*** | **210** |

### R4.2.3 – Security of Sensitive Transmitted Information

**Evaluation**

By default XOOPS does not use a secure connection for transmission by default, but it has the ability to accept connections through SSL, if set up and configured that way. This requirement can be met by setting up an SSL page for use with user logins, then configuring XOOPS to use this page. Making the configuration change is simple since XOOPS has an option for using SSL logins. Most of the effort needed to meet this requirement would be in constructing the SSL page for use by XOOPS.

**Subsystems Requiring Modification**

- **Configuration:** installation with https support

**Complexity** 1/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| Configuration | 10 |
| ***Total*** | **10** |

### R4.2.7 – Compliance with Other Technical, Accessibility, and Security Requirements

**Evaluation**

The system follows some, but not all of the policies for NASA web sites, such as the standard NASA header and footer information.

**Subsystems Requiring Modification**

- **PHP frontend:** modify files for display
- **Layout:** add header and footer information

**Complexity** 2/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| PHP | 50 |
| HTML | 100 |
| ***Total*** | **150** |

### R4.2.8 – Policies Availability to Users

**Evaluation**

Not supported. There is no specific feature available for making policies available to users. However, it can be met easily by providing the policies or links to them somewhere on the XOOPS site, in an appropriate section and clearly labeled. The work necessary to meet this requirement is minimal.

**Subsystems Requiring Modification**

- **Layout:** add documentation to site using tools

**Complexity** 1/10

**Estimate Lines of Code**

| ***Language*** | ***Lines of Code*** |
|---|---|
| HTML | 100 |
| ***Total*** | **100** |

### R4.3.1 – Function as a Repository

**Evaluation**

Not supported. The default downloads module does not have the ability to accept uploaded files. This feature is provided by at least one module in the repository at the official XOOPS site, so installing this or another module with similar functionality would be necessary to meet this requirement.

**Subsystems Requiring Modification**

- **Configuration:** install upload module

**Complexity** 1/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| Configuration | 10 |
| ***Total*** | **10** |

**R4.3.3 – Selection of System Behavior by Provider**

**Evaluation**

Not supported. This feature is provided by requirement 4.3.1 (Function as a Repository).

**Subsystems Requiring Modification**

- **PHP frontend:** UI for providers to choose how they want to store an asset.

**Complexity** 1/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 100 |
| ***Total*** | **100** |

**R4.4.1 – Asset Deprecation by Content Managers**

**Evaluation**

Not supported. There is only an option to remove/delete assets, in fulfillment of Requirement 4.4.2. There is no option to keep the asset in the system, but not display it publicly. Content Managers may be able to mark or otherwise identify which assets should be deprecated and deleted by Administrators, but this alone would note deprecate the asset. We would have to code this feature.

**Subsystems Requiring Modification**

- **PHP frontend:** UI for toggling deprecation flag with appropriate permissions
- **Database:** add field for deprecation flag

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 100 |
| SQL | 5 |
| ***Total*** | **105** |

### R4.5.1 – Verification of Data by Providers

**Evaluation**

Requirement 4.5.1, Verification of Data by Providers, is partially met because information such as a checksum can be included in the asset description. There is no dedicated feature for this, however.

**Subsystems Requiring Modification**

- **PHP frontend:** add checksum fields when submitting new assets, perform check
- **Database:** add field to store checksum and checksum type

**Complexity** 3/10

**Estimate Lines of Code**

| *Language* | *Lines of Code* |
|---|---|
| PHP | 200 |
| SQL | 5 |
| ***Total*** | **205** |

## Appendix G – Glossary of Terms

- Administrator – a user who controls, operates, and manages the system
- Asset – an item produced at some point in the software development life cycle that is recognized as having a particular value
- Catalog – a system that stores links to assets, but does not store/host the assets themselves
- Consumer – a user, either registered or unregistered, who is allowed to access or otherwise use assets in the system, subject to their license terms
- Content Manager – a user whose main role is to review content submitted to the system (e.g., a new asset) for appropriateness and relevance
- Portal – a system that serves as a single point of access to varied information and provides a consistent look and feel for accessing that information
- Provider – a registered user who has been granted permission to upload asset resources and metadata to the system
- Registered user – a user who has completed a registration process in order to obtain an account on the system
- Repository – a system that stores/hosts the actual assets themselves
- Submit – refers to the process by which information is provided to the system for inclusion in the system
- Unregistered user – a user who has not completed a registration process in order to obtain an account on the system
- User – any person who accesses the system
- Web site – a collection of Web pages (documents), images, etc. available on the World Wide Web