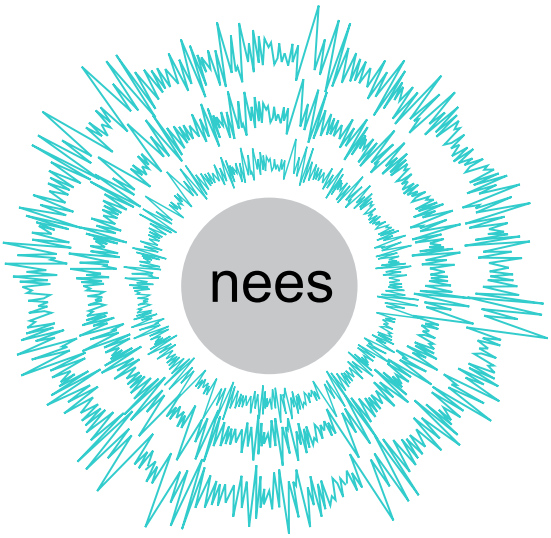

NEESgrid SYSTEM ACCEPTANCE TEST PLAN

Revision 1.0

January 31, 2004

This work was supported by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Numbers CMS-0126366 and CMS-0117853.

PREFACE TO REVISION 1.0

Acceptance testing is perhaps THE critical step of a software development effort. It is a formal process whereby the performance, appearance, and usability of the software are measured and compared to criteria agreed upon by the developer and the user/client. Acceptance testing is intended to confirm that the software system and its components meet the specifications formulated as part of the development process.

This draft Acceptance Test Plan for NEESgrid has been developed by the NEES Consortium Development Project and NEES Consortium, Inc. (the final "user/client" of the NEESgrid software system) in collaboration with the NEES System Integration team. It is intended as a starting point for formalizing a mutually agreeable acceptance test plan, procedures, schedule, and execution. The ultimate goal is NEES Consortium, Inc. acceptance of the NEESgrid system and its components as a product that will be useful to the Earthquake Engineering community.

There is no formal system and software requirements document for NEESgrid, nor clear specifications of the NEESgrid components. This is a result of the somewhat inverted nature of this development effort – where the client/user (the Consortium) did not exist when the development effort began. As a consequence, the authors based the requirements section of this plan on the document entitled "NEESgrid System Overview" (by Tom Prudhomme, Version 2.1, dated Oct. 30, 2002). The system's functional requirements are extracted from the text beginning with the section "Collaborative Project Design and Planning" (page 5) and ending with "Data Repository" (page 11"). The SI team defined the system components it has developed and described how these components map to the functional requirements

This version of the Acceptance Plan lays out the system requirements, the components that are to be delivered, and the procedures for how those components will be

evaluated for acceptance. Detailed information on test procedures will be added as appendices. It also demonstrates how the specific functionality criteria to be tested will be derived from the general text of the System Overview document.

It is hoped that the formalization of an Acceptance Test Plan and Procedures will help ensure the delivery of an excellent software system that contributes to the success of the Consortium and of the NEES program in general.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Background

The Systems Integration (SI) project within the NEES MREFC effort is responsible for developing the IT system for NEES. That system, called NEESgrid, will be completed by September of 2004. The system will be delivered to NEES Consortium, Inc., the entity that will manage NEES during the operational phase from 2004-2014 and beyond.

This Acceptance Test Plan formalizes the expected handoff of SI products to the Consortium. It provides a mechanism for the Consortium to review and accept the results of the SI efforts and to put formal closure on that portion of the Cooperative Agreement between the SI and NSF.

The goal of the acceptance testing described in this document is to verify the overall quality, correct operation, scalability, completeness, usability, portability, and robustness of the functional components supplied by the SI team..

1.2. Software System Requirements

In the absence of formal Software System Requirements and specifications documents within either the SI or the Consortium, the SI document entitled “NEESgrid System Overview” (Version 2.1, dated October 30, 2002) is used as the basis for development of this Acceptance Test Plan. Functional requirements for various components of the NEESgrid system have been extracted or modified from that document and input from component and team leaders with the SI group. Table 1 presents those functional requirements, organized into general functional categories and expressed in the words used by the System Overview document.

It is important to note that these are requirements of the NEESgrid system taken as a whole. Section 1.3 details the NEESgrid components developed by the SI team and explains how each component maps to the requirements in this section.

Table 1.

System requirements (subdivided by functional category)

Group A: Collaborative Project Design and Planning

General Description	The first stage in the research process is to take an idea and develop it informally through discussions with colleagues to formulate a study topic and research question. The next step is to discover and document what is known about the topic, including reviewing published research results and data that could be used to formulate hypotheses relating to the research question. Then, building upon what is known, the researchers decide upon a unique study that would add to the body of knowledge in the field, and begin the process of project planning. Typically, this process includes deciding with whom to collaborate, what facilities or other resources are required to conduct the study, how long it will take, and what it will cost. NEESgrid supports this phase of the research process with specific tools for data discovery, collaboration and data viewing.
A.1 Repository Search and Discovery	The metadata catalogs for the data repository and community simulation code repository are searchable against any of their fields, and a simple search tool is provided to help end users locate any testing information or information about simulation codes or modeling tools relating to the research problem. Locating published testing data and all associated metadata allows researchers planning new projects to evaluate every aspect of another study that was done independently by someone else at any one of the NEES facilities. Other information that is discoverable includes simulation codes or modeling tools (from the community simulation code repository), other NEESgrid participants with similar research interests (from the www.nees.org member database, and from the metadata catalog), and NEES facility availability and capabilities (for NEES Equipment Sites supporting online scheduling and inventory discovery).
A.2 Online Collaboration	Support for online collaboration under NEESgrid is mediated by the Worktools and CompreHensive collaborativE Framework (CHEF) developed at the University of Michigan. These environments include a variety of resource management and information sharing capabilities. In addition to supporting asynchronous activities, the NEESgrid collaboration environment will support synchronous, or real-time collaboration through text chat, information sharing and videoconferencing. Multi-site videoconferencing sessions will be

	scheduled on NEESgrid using multipoint control units dedicated to that purpose that are shared by all NEES users. Videoconferencing sessions will require Polycom™ systems (or fully compatible H.323 systems) at each site, and can be conducted by themselves or in association with other collaboration tools. The NEESgrid collaboration environment also includes an electronic laboratory notebook, which in addition to these tools can be used to document the planning and design process.
A.3 Data Analysis and Visualization	The data repository and metadata catalog will contain information about both physical testing studies and numerical simulations that have been conducted as part of the NEES program. The NEESgrid collaboration environment will support viewing this data and metadata (as described in requirements groups C and E) so that scientists and engineers may review past experience in the community.

Group B: Setup, Observation and Monitoring of Tests at NEES Sites

General Description	For research projects that require physical testing at NEES Equipment Sites, the next phase of the project is preparation, which can take many months. Resources need to be scheduled, specimens designed and built, and equipment and sensors calibrated. Testing is complex and expensive, and must be completed within a narrowly defined time window. Thus, when tests are run they need to be monitored in real time (or near real time) to evaluate preliminary results and determine if the tests were successful. NEESgrid supports these steps in the research process with the CHEF tools as well as an electronic laboratory notebook, a telepresence system, and its collaboration and data-viewing environment.
B.1 Electronic Laboratory Notebook	The NEESgrid laboratory notebook will help the researcher capture some of data during the preparation process. These data might include notes, equipment or sensor calibration data, and video captures of the specimen preparation process and placing of sensors. This process will not be automated, but it will be under the direction of the researcher. When the tests are run, preliminary results can also be included in the lab notebook as additional notes, still video captures, and/or simulation models images, again at the discretion of the researcher. The laboratory notebook is not intended to be a data archive, but will be a rich source of metadata that will help people uninvolved in the study to understand and evaluate that study for their future use. Examples of typical data that might be included in an e-Notebook are experimental notations, calibration data, data plots, and sample data tables.
B.2 Telepresence System	The NEESgrid telepresence system (TPS) supports tele-observation. Tele-observation is focused on real-time and near real-time viewing of multiple data and streaming jpeg (video images) during tests conducted at NEES Equipment Sites. All streaming images are coordinated using a Linux based TP Server, while to achieve the performance required for

	viewing sensor data during tests, the experimental sensor data are processed and streamed to clients from the NEES-Pop using a CHEF interface. In addition, sensor data is simultaneously saved on the data acquisition system. Additional data may be uploaded from the data acquisition system to the local data cache during the trial; at the end of the trial, data from all sensors are uploaded to the local data cache. This approach allows real-time viewing to be coupled with subsequent playback/review and posting of the data to the repository when the testing is completed using CHEF tools. Saving all sensor data on the data acquisition system while the test is running also allows for the test to run without loss of data in the event of a failure of the streaming data server or the network connecting the data acquisition system to the NEES-POP.
B.3 Data Streaming	Remote observation of experiment results requires the streaming of experimental sensor data as it is being captured by the equipment site data acquisition system. It must be possible for a client to subscribe to one or more data streams, and correspondingly it must be possible for the equipment site to control access to data streams, to be able to generate "virtual" data channels (to accurate server side data filtering, and data reduction operations).
B.4 Data-viewing Environment	Experiment data and metadata viewing tools (described in requirements group E) must be accessible during and immediately following an experiment so that users may determine whether or not the experiment was successful. The viewing perspectives can include numerical data, graphical representations or video data, and all perspectives are time synchronized.
B.5 Telecontrol	The NEESgrid Telecontrol Protocol and Service (NTCP) supports the control of physical experiment equipment and computational simulations by remote applications. NTCP provides a common protocol to communicate with experiments running on heterogeneous hardware and software platforms and APIs to be used for integration with remote client processes and local control system (or simulation) interfaces.

Group C: Using the NEES Data Repository

General Description	Once the testing component of a project is complete, the test data and all associated metadata are posted to the NEESgrid data repository for further analysis and for use by all authorized users. The NEESgrid Data Repository is an important grid resource for supporting collaborative research involving testing. It supports access controls to limit use to the project participants, but each participant can access the data and metadata via a common interface. Once the study is completed and the data are released by the original owner(s) for general user access, the data repository becomes an invaluable resource for other researchers modeling phenomena based on the results of multiple independent tests

	or developing new research hypotheses based on rectsults from prior studies. The key element in a data repository is the metadata catalog. The metadata define the data and the circumstances surrounding their collection. They describe the numerical data formats and the sensors used in collecting them. They describe still and motion video data, the cameras used, and the formats for viewing the resulting images. The NEESgrid team has developed a metadata model that should ultimately serve as a community standard for specifying metadata in the repository. The general approach is to capture ("ingest") essential information about a test (i.e. the metadata, as distinct from the data) from all the sources described earlier (i.e. chat sessions, the electronic laboratory notebook, structured metadata entries, and other NEESgrid resources described earlier. This, coupled with other information provided by the owner of the data, completes the detailed and documented information describing how a test was conducted and its outcome. The benefits of community-wide agreement on a metadata model are facilitated data discovery and access by (a) members of the technical field that conducted the tests, and (b) members of other technical fields who may also be interested in the results. This feature is key to realizing the NEES vision of promoting cross-domain experimental or numerical simulation research activities between, e.g., structural and geotechnical earthquake engineering. The establishment of consensus-based community data standards and specifications is a complex task. The approach taken by the NEESgrid team is to agree upon a solid and usable data model specification for October 1, 2004, based on detailed interaction with the community, its research use scenarios, and available model data sets from research conducted using different NEES facilities. This approach provides a framework for a community-based process for agreeing upon and documenting modifications or extensions to the standard data model during the operational phase of the NEES Collaboratory (2004-2014). This process will be managed by the NEES Consortium, and will provide important flexibility to accommodate changes in the modes and types of earthquake engineering research conducted by the NEES community. The data repository established during the integration phase of NEES is built upon on a rich set of APIs that makes it possible for future NEESgrid users to develop powerful custom interfaces to NEES data and metadata. Certain reference implementations of tools and user interfaces are provided as part of the integration phase.
C.1 Metadata Ingestion Tools	Primary sources of metadata describing a test conducted at a NEES Equipment site are the electronic notebook, structured metadata, and additional free form notes provided by the experimenter. Metadata ingestion tools provide the capability to excerpt information from the electronic notebook and use it to populate the metadata model. Metadata is also developed in highly structured formats such as XML-based formats, and ingested into the repository though format

	translation. The SI will provide a reference implementation of data ingestion for a set of example formats. A simple free-text web form is provided to capture additional comments from the PI or other members of the research team and insert them into the metadata model.
C.2 Data Discovery and Access Tools	These tools facilitate finding physical testing or simulation data (and all descriptive metadata) of interest to a researcher that was previously published by a different researcher. They allow users to search any element or combination of elements in the metadata catalog, and access the data for collaborative viewing or analysis in the CHEF environment.
C.3 Curation Management Tools	The data repository is curated, and tools are provided to ensure that all information is organized, classified, indexed, moved, linked, annotated, versioned, archived, and secured. The tools represent an integration of best-of-breed collection-management tools augmented with specialized tools for specific data, as required. The tools support the activities of the repository curator, who will be responsible for verification and validation of data published in the repository.

Group D. Using the Numerical Simulation Tools and Repository

General Description	NEESgrid will facilitate researchers' remote shared access to experimental equipment and data, will facilitate efficient communication between researchers, and will provide a powerful collaborative environment for modeling and simulation. This section describes NEESgrid components supporting research based on computational simulation techniques. The NEESgrid simulation effort includes several components supporting the goal of consolidating computational simulation tools from the earthquake engineering community into a community code repository of tested and capable software tools. The simulation effort is oriented towards incorporating existing software in the earthquake engineering field into NEESgrid as a means of improving understanding, accessibility, and use of these tools by a broader community of earthquake engineers. NEESgrid simulation activities include gathering of community tool information, development of a community software repository, demonstration and documentation efforts on selected software tools, and identification and implementation of enhancements for key applications in the support of improved NEESgrid system usability. The grid components of the system will support secure single login access to a variety of high performance computing and storage systems available to grid users, for example through the National Center for Supercomputing Applications. There are no guarantees that NEES researchers will have access to these facilities – they must request allocations through the normal channels – but the NEESgrid

	collaboration environment allows simulation code to be run using Grid mechanisms to involve external computational and storage resources. For example, a reference implementation will demonstrate the system calls needed to transfer data from the NEES repository to a TeraGrid partner site and vice versa.
D.1 Community Code Repository	The NEESgrid team will design, develop, and deploy a software repository of selected simulation tools for earthquake engineering applications. Metadata will be provided about the simulation software functionality, platform, and verification/validation criteria, so that the data repository's browsing and search functions can be used to locate appropriate simulation software.
D.2 Reference Documentation	The NEESgrid team will develop reference documentation for the community software repository to be used a means to educate and orient NEESgrid users, engineering practitioners and the general public on the use of the repository.
D.3 Portal Interface	The NEESgrid team will design and develop community-motivated software enhancements that will add value to representative community codes, e.g., developing a portal interface for an earthquake engineering simulation tool such as the OpenSEES framework (opensees.berkeley.edu).

Group E. Using the CHEF Framework for Analysis and Sharing of Results

General Description	A NEES project can generate large quantities of text, graphics, video and numerical data. NEES users need a framework for accessing, analyzing and sharing these data and analytical results. NEESgrid provides a collaborative environment customized to serve the needs of the NEES user communities. The collaborative framework is built upon the WorkTools environment developed at the University of Michigan, extended into a general purpose comprehensive collaborative framework called CHEF. The CHEF environment is used in the NEESgrid collaboratory to allow earthquake engineers to configure a virtual workspace to fit their particular needs. This includes selection of specific collaboration and research tools, such as electronic lab notebook, data archive search, email archive, or visualization tools, and specification of the appearance of these tools within the collaborative workspace. In addition to the collaborative workspace, individual engineers have their own personal workspaces that they can customize to meet their own needs. A key strength of the CHEF approach is the reliance on open source technology and standards so that additional tools can be quickly integrated with existing tools.
E.1 Web-based Collaboration	These tools are customizable by users and support scheduling, announcements, task management, information resource management, discussion spaces, shared presentations, notification services, email

Tools	group management, email archives, and laboratory notebooks.
E.2 Grid-enabled CHEF extensions	These extensions support security/authentication (useful when signing on to multiple secure resources), resource discovery (e.g., from the data repository or other data resources on NEESgrid), and directory services (e.g., for locating collaborators) for users within the collaborative environment.
E.3 Data Viewer	The data viewer is designed to provide time-synchronized viewing of data from multiple perspectives, and representing multiple data types. This tool takes inputs from repository data; the user selects the experiment, the channels (data and/or video); the user chooses plot types; and the results are displayed in a tiled format showing time-synchronized plots of selected channels (or time synchronized video).
E.4 End User Application Integration	The CHEF-based, NEESgrid collaborative environment supports basic and advanced application integration into the framework. Any existing application that is in the form of a web page or a web application can be easily incorporated into a personal end user space or shared with other collaborators. The CHEF system provides well-defined APIs and service interfaces—along with corresponding documentation—to allow the integration of advanced applications using multiple NEESgrid services and resources.

Group F. Use of NEESgrid by Practitioners

General Description	Practitioners are frequently engaged in research and testing activities with earthquake engineering researchers. In addition, they may want to review existing data and other information resulting from similar tests conducted by non-participating researchers. Finally, they may want to consult with an academic engineer while simultaneously viewing such data. NEESgrid will support all of these uses by practitioners through the tools other capabilities described in previous sections.
F.1 Telepresence Systems	Collaborating practitioners (e.g., from a state Department of Transportation) can remotely view tests conducted at NEES Equipment Sites while collaborating with their research partners to view the streaming data (from sensors and video sources) and quickly analyze the results in near real-time (CHEF).
F.2 Electronic Lab Notebook	The electronic lab notebook must allow users to enter an experiment's entire design, specimen preparation, and test setup process.
F.3 Data Repository	Discovery tools in the repository coupled with CHEF collaborative tools give practitioners the ability to identify and review similar or otherwise relevant results from prior tests for which they have access privileges.

Group G. Managing and Supporting NEES Collaboratory Operations

General Description	Underlying the tools and interfaces described in this document is a sophisticated middleware environment architected to manage the interactions among resources and mediate the interfaces between end users and the resource grid. These resources include, e.g., the NEES Equipment Sites, the Data Repository, and various computational resources accessible by NEESgrid users. The details of the system architecture are available at www.neesgrid.org/NSFreview/ NEESgrid_SA_Feb15.2002.pdf; the system architecture specification document describes the grid services and protocols being used in the "background" to make it easy for users to conduct their research and other work. Scenario-based applications of NEES-POPs at the early adopter sites are described in the Early Adopter Plan document also located in the neesgrid.org library.
G.1 NEES-POP	In order to facilitate end user access and use of all the NEES Equipment Sites, a standard interface was designed to support the grid services specifically enabling the telepresence, collaboration, and data repository interfaces described above. This interface is called a NEES Point of Presence (NEES-POP), and the common services that reside on each NEES-POP are customized to the unique requirements of each Equipment Site. The end result is that the experience accessing and using one NEES Site is very similar to the experience using another.
G.2 Operations Center and Central Services	A critical component of the physical dimension of the overall collaboratory is system performance, including all the NEESgrid computing, storage and networking equipment, and the national broadband Internet-2 network that ties it all together. During the SI contract period, the SI team has provided an Operations Center that monitors the performance of all aspects of the NEESgrid and the national broadband network. The Operations Center maintains a help facility that can quickly diagnose and respond to user-generated requests. In addition to traditional operations center tasks, centralized services essential to NEESgrid operations provide capabilities needed for end users as well as system administrators. These services include the capability to issue X509 certificates for NEESgrid authentication, account management tools for user accounts across NEESgrid sites, and a credential repository for storing long term credentials needed for authentication using CHEF. Responsibility for providing these services will transition to the Consortium upon completion of the SI contract. To facilitate this transition, the SI has developed descriptions of each of these services including descriptions of any open source or off-the-shelf tools used by the SI to implement them.
G.3 Training Materials	Learning to use the physical NEES resources using remote interfaces is an important component of the NEESgrid environment. During the final year of the integration phase, the NEESgrid team will conduct user training workshops to ensure that engineers know how to take full

	advantage of the features of the NEES Collaboratory. The training materials will be made available to NEES Consortium for on-line use by all users for the life of the NEES Program.

1.3. NEESgrid Components

This section details the NEESgrid components developed by the SI team and explains how each component maps to the requirements defined in Section 1.2.

1.3.1. Streaming Data Components

Streaming data components allow remote applications to receive streaming data from experiments running within NEESgrid. This supports system requirement B3 in Section 1.2.

1.3.1.1. Neesgrid Streaming Data Service NSDS Service

The NSDS service runs on a NEESpop server at an Equipment Site. It provides a standard network service interface to the (near real-time) streaming data generated by the site-specific data acquisition (DAQ) system.

Unit tests for NSDS will involve development of test drivers and clients; the test driver will stream known sets of data points to known channels; the test client will subscribe to channels and record the data streams received to files. The results (source data and received data) will then be compared. Since NSDS makes a best-effort attempt to deliver data in a timely manner, some (yet to be specified) percentage of data loss will be considered acceptable. . Please note that NSDS is not the mechanism by which data is ingested into the data repository. The data repository gets ALL the data from the experiment (through the DAQ). NSDS clients observing the experiment (viewing some fraction of the “near live” data) may not see all of it, due to the best-effort basis.

Work Flow test: Stream data from a driver through NSDS and view in the data Viewer (part of visualization efforts)

1.3.1.2. NSDS Client Libraries/APIs

Administrative Client: Invokes the administrative interface of the NSDS

Regular Client: Invokes services provided for the users of NSDS.

Unit Tests: See 1.3.1.1.

1.3.1.3. NSDS Drivers and Plug-ins

Driver Interface: Handles connections to drivers and sends appropriate commands on behalf of the NSDS server.

1.3.2. Teleoperation Control Components

Teleoperation in NEESgrid is implemented using this model: *client applications* (simulations) make requests using the NEESgrid Teleoperation Control Protocol (NTCP). *NTCP servers* receive these requests, process them, and forward them on, via the use of *plugins*, to *backend systems* (control systems or simulations). A *plugin* is a small piece of code specific to the particular backend system (hardware controller or simulation) in use at a site.

1.3.2.1. NTCP Server and Client API

The core NTCP server which, when used with a "plugin", is used to communicate with a backend control system (or simulation). In the MOST experiment, three NTCP servers were used: one at NCSA to handle communications with the simulation at NCSA, one at Colorado to handle communications with the Matlab application there (which in turn used xPC to control servo-hydraulics), and one at UIUC to handle communications with the Shore Western servo-hydraulic hardware there.

1.3.2.2. NTCP Java Client Library

A java class used by client applications to communicate with NTCP servers. In the MOST experiment, the coordinating simulation used the NTCP java client library (indirectly, via the NTCP Matlab client library) to communicate with the NTCP servers running at UIUC, NCSA, and Colorado.

1.3.2.3. NTCP C Client Library

A set of C functions used by client applications to communicate with NTCP servers. This performs the same function as the NTCP java client library, but provides an interface in the C programming language. (Note: this hasn't been written yet).

1.3.2.4. NTCP C Gateway Plugin

An NTCP plugin, written in java, that calls a set of C callback functions to communicate with some backend control system (or simulated control system) using a C API. We will define and publish this C callback API; a set of functions that implements this API to communicate with a control system (or simulated control systems) is called an "NTCP C plugin". So the C-gateway plugin is an NTCP java plugin that calls an NTCP C plugin, which in turn talks to some control system. To test the NTCP C-gateway plugin, we'll also have to provide a "dummy" C plugin (just as we included a "dummy" java plugin for testing the NTCP server).

1.3.2.5. NTCP Mplugin

An NTCP plugin, written in java, that is used to communicate with Matlab backend simulations. In the MOST experiment, the Matlab backend components (the simulation at NCSA and the application at Colorado) communicated with NTCP via the Mplugin.

1.3.2.6. NTCP Matlab Client Toolbox

A set of Matlab functions that call the NTCP java client API to communicate with an NTCP server (I think the implementation of this may also include some java functions in between the NTCP library and the Matlab functions). In the MOST experiment, the coordinating simulation used this library.

1.3.2.7. NTCP Matlab Backend Toolbox

A set of Matlab functions that call the Mplugin java client API (the Mplugin actually implements its own grid service) to communicate with the NTCP server running on

the local NEES-POP. In the MOST experiment, the simulation at NCSA and the application at Colorado used this library.

1.3.3. Data Acquisition Subsystem

The data acquisition (DAQ) subsystem provides tools and components for sites to integrate their laboratories into the NEESgrid. Included are a reference implementation in LabVIEW, and an API for them to use in their DAQ code. Also available are implementations in C for two other inexpensive DAQ devices, and a pure software implementation for use as example code.

1.3.3.1. Data Acquisition Service

The DAQ code runs two programs on the sites' DAQ computer: The site-specific DAQ code, and a small server daemon to handle the TCP/IP connections to the driver. The server daemon manages the channel list (subscriptions), connection establishment and reestablishment, status propagation, and all remote queries. The DAQ code is written such that it can run even if the server daemon is not present or the network is down; connection handling and such are handled behind the scenes so as to minimize our impact on their experiments.

While the reference implementation is in LabVIEW, others are included that speak the same protocol. We have written a tool that does an exhaustive test of protocol compliance, correctness and error handling; this will be posted to the web for sites to check their implementations as well as our own testing.

1.3.3.2. DAQ Client Library/API

The LabVIEW API provides functions to read and write metadata to disk, stream data to the driver/NSDS, perform repository uploads via FTP, save to disk, set DAQ status, etc. It is designed to be added to existing site-specific DAQ code with minimal effort and impact.

Code is included with the LabVIEW DAQ distribution to test its correctness and scalability, and can be run at will at any site. As an aid, we also supply a compiled NSDS simulator that plots the streaming data for sites not possessing a LabVIEW license.

1.3.3.3. NSDS Driver

The NSDS driver is a plug-in based driver model between the DAQ and the NSDS. Our DAQ test tool will also handles testing for the driver.

1.3.4. Data Management Services

NEESgrid data services provide the ability to ingest, manage, retrieve, and archive data and metadata in a secure, collaborative environment. Data services are provided as a set of core OGSI (Open Grid Services Infrastructure) services providing applications with the ability to manage data and metadata objects and resources. In addition, a set of reference end user applications have been built on those services. The Open Grid Services Infrastructure (OGSI) is a set of WSDL specifications defining standard interfaces, behaviors, and schema for grid computing consistent with the OGSA. OGSA, which stands for Open Grid Services Architecture, provides standard communication protocols and formats. OGSA represents the means to build truly large-scale, interoperable grid systems. NEESgrid services are built upon this framework.

Data services are packaged as part of the data and collaboration package. Testing of the packaging process for NEESgrid data tools will be accomplished in the context of testing the data and collaboration package(s).

The NMDS implementation provides a suite of unit tests that verify API correctness as well as metadata repository implementation behavior under a variety of error conditions as well as ideal conditions. The test is run in two stages, first directly against NMDS's JDBC (Java DataBase Connectivity) implementation (including the Mysql-specific features), and then against the web service implementation of the same API.

Another set of tests evaluate NMDS performance under various load volumes both in terms of rate of access and number of simultaneous clients. These extend the tests used for the preliminary performance analysis completed in Spring 2003.

1.3.4.1. NEESgrid Metadata Service

The NEESgrid metadata service (NMDS) implements repository functions associated with metadata, including metadata storage, retrieval, search, security, and provision of an API. This supports CHEF-based metadata ingestion, browsing, and search end user capabilities, as well as supporting archiving and application access to metadata. It also provides the file management system (described below) with a way to store and retrieve metadata about files and file transfers.

The NEESgrid Metadata Service (NMDS) and NEESgrid File Management service (NFMS) implement data repository functions. API correctness tests will be included for both successful and error conditions. In the case of NMDS, the test suite is run in two modes, 1) direct JDBC access to a test repository database and 2) remote, web-service access to the same test repository database. The database is a Mysql database configured the same way as it is in the release. NFMS uses NMDS, so the successful completion of its test suite depends on successful completion of the NMDS test suite.

1.3.4.2. File Management Service

The NEESgrid file management service (NFMS) implements repository functions associated with data, including sensor data, images, video, and documentation. These functions include security, file transfer, naming and directory services, linking files to metadata, and provision of an API. This supports CHEF-based data ingestion, browsing, and data viewer interfaces.

The NFMS implementation provides a suite of unit tests that verify API correctness as well as NFMS implementation behavior under a variety of error conditions as well as ideal conditions. Another set of tests evaluates NFMS performance under various load volumes both in terms of rate of access and number of simultaneous clients.

1.3.4.3. Repository Browser Teamlet

The NEESgrid repository browser teamlet provides a user interface to the NEESgrid repository. It allows the user to upload, download, browse, and search data and metadata.

A set of test scripts walk a tester through an exhaustive set of scenarios for the browser, exercising all of the user interface features and providing a description of the expected behavior at each point.

1.3.4.4. Metadata Ingestion Tool

The NEESgrid metadata ingestion tool provides a means for uploading metadata to the repository in one of the supported metadata formats. It includes a suite of tests that not only verify correct behavior under error conditions, but also upload test metadata, re-download the metadata, and check that the correct metadata has been stored in the repository. This is a multi-step process which tests every attribute of every object to make sure that the metadata is correct. A set of test metadata is provided that exhaustively exercises the repository's object-oriented model of metadata structure.

1.3.4.5. Central Archiving Service (“central repository”)

The NEESgrid central archiving service implements repository functions associated with long-term preservation, including mirroring of data and metadata as well as tertiary storage strategies. This central repository will be able to harvest data and metadata from sites and move it into the tertiary storage environment while maintaining the ability to securely retrieve the objects. A set of automated tests will verify that data and metadata can be retrieved at every stage in the archiving process without any loss of data.

1.3.5. CHEF and Worktools

CHEF (www.chefproject.org) and Worktools provide an online collaborative environment as part of the NEESpop distribution. CHEF is developed and supported by the University of Michigan.

1.3.5.1. Chef Base Distribution

The CHEF Base distribution provides a number of capabilities "out of the box" including (1) Schedule Tool, (2) Threaded Discussion Tool, (3) Flat Discussion Tool, (4) Live Chat Tool, (5) Resource Tool, (6) an I-Frame tool (capable of displaying web pages), and (6) a set of site administration tools.

These tools are tested as part of each release of CHEF by University of Michigan personnel. For each version of CHEF which is integrated into the NEESGrid toolset, the University of Michigan will provide a copy of the testing report for that version of CHEF.

1.3.5.2. Chef Grid Services Package

The Chef Grid services package adds the following components: (1) Grid Certificate Display Tool, (2) Java CoG based GridFTP (adapted from Indiana University), and (3) An application Service which provides CHEF applications to Grid credentials.

1.3.5.3. Chef MySQL package

CHEF supports MySQL as part of the base release. As part of the acceptance testing for each release of CHEF which is integrated into NEESGrid, the standard CHEF test scripts (1.3.5.1) will be run on an installed CHEF configured to use MySQL.

1.3.5.4. NEESgrid Visualization Tools

The University of Michigan also provides two visualization tools: (1) stored data visualizer which retrieves data from the data repository and (2) a streaming data visualizer which interacts with the NSDS Streaming data protocol. The stored data visualizer can display several different types of channels including: (1) stored video

(only formats supported by the Java Media Framework), (2) scalar floating point data, and (3) two-dimensional structural analysis model data. The stored visualizer shows multiple synchronized streams at the same time in its interface, allowing for the synchronization of video and data. The streaming viewer only supports scalar channels. The NEESpop as shipped in version 2.0 will include sample data both streaming and stored which demonstrates all of the capabilities of the tools. This sample data is also provided and will be used as part of the testing script for these tools.

1.3.6. Telepresence Video Components

The functionality and test procedures for TelePresence Video will be provided as a script that testers can follow, walking them through a specific scenario of logging in, and opening the various options associated with near realtime imaging from a TP site, it will exercises every interface feature and contains specific instructions for logging errors.

In general, functionality testing for this group of NEESgrid software services is accomplished simply by logging into services and verifying that the requisite objects are available or present. For example, to test streaming video the evaluator will simply login to the TP server using a WWW browser and confirm the presence of a streaming jpeg image in their browser window. In addition, custom programs are provided to measure and calibrate system performance from the TP Administrative Console. A script for these actions will be included as part of the tests described above.

1.3.7. Electronic Notebook

The electronic notebook is a component which allows experimenters to capture and share data during the experiment preparation phase, experiment running phase, and post trial phase akin to a written laboratory notebook. Data might include notes, equipment or sensor calibration data, and video captures of the specimen preparation process and placing of sensors.

1.3.7.1. E-Notebook Unit Test

The functionality and test procedures for the Electronic Notebook will be provided as a script that testers can follow, walking them through a specific scenario of logging in, creating entries, appending information, uploading and downloading files, storing frame captures from the TP Video systems etc. that exhaustively exercises every interface feature and contains specific instructions for logging errors.

In general, functionality testing for this group of NEESgrid software services is accomplished simply by logging into services and verifying that the requisite objects are available or present. For example, to test capturing a video image and annotating the evaluator will simply login to the E-Notebook using a WWW browser and select the appropriate option, fill out the metadata annotation form, select the video source and confirm the creation of the entry in the E-Notebook in their browser window.

1.3.8. Simulation Services

The simulation services will be built upon the Open System for Earthquake Engineering Simulation (OpenSees). OpenSees was developed by the Pacific Earthquake Engineering Research Center as an open-source software framework for simulation of structural and geotechnical systems. It has the capability for nonlinear (material and geometry) static and dynamic analysis using a variety of beam-column and continuum elements and a library of material models. OpenSees is a modular, object-oriented software that is extensible for providing new types of numerical models (e.g. elements and materials), analysis procedures, and interfaces (such as to databases and NTCP).

1.3.8.1. Community Software Repository

A software repository for OpenSees will be created to allow for continued development of the framework by the earthquake engineering community. NEESgrid facilities will be utilized for a community-based simulation software development process, including a roadmap, project and bug tracking systems, user mailing lists, discussion forums, and processes for submission of material from the community. Browsing and searching for OpenSees API's, source code and

documentation, including examples and tutorials will be supported by the data repository's functionality.

1.3.8.2. Portal Interface

The NEESgrid simulation services will include a portal interface for community software in the code repository using the NEESgrid portal framework. The portal allows for secure interactive job submission, batch job submission, status and monitoring, and a service to configure a simulation execution.

1.3.8.3. Metadata Definition and Data Interfaces

A data schema for simulation models and simulation results will be used for archiving in the NEESgrid data services. The OpenSees API will be extended to provide interfaces with the NEESgrid data repository for models and simulation results.

1.3.9. Centralized Services

As part of NEESgrid, we operate servers that provide live information about NEESgrid, including health and Status (based on Big Brother Monitoring), NEESgrid wide centralized Information Server (GIIS server), custom information providers, and access to a Certification Authority etc. Please see write up in Table 1, Section G.2.

1.3.10. How Components Map to Requirements

Table 2 shows how the components listed earlier in this section relate to the system's functional requirements from Table 1.

Table 2. Components Mapped to Requirements

Component…	Meets Requirements…
NSDS Service	B.2, B.3, F.1
NSDS Client Libraries/APIs	B.2, B.3, F.1
NSDS Drivers/Plug-ins	B.2, B.3, F.1
NTCP Service	B.5

NTCP Client Libraries/APIs	B.5
NTCP Drivers/Plug-ins	B.5
DAQ Code	B.2
DAQ Library APIs	B.2
Metadata Service	A.3
File Management Service	B.2
Repository Browser teamlet	A.3, B.2. B.4, D.1
Metadata ingestion tool	A.3, C.1
Central archiving service	A.3, B.4, C.1,C.2, E.2, F.3
Chef Base Distribution	A.2, C.2, E.1, E.2, F.1
Chef Grid Services	A.2, C.2, E.1, E.2, F.1
Chef MySQL	A.2, C.2, F.1
Data Visualization Tools	A.3, B.2, B.4, C.2, E.3
Telepresence Components (NTCP, NSDS and CHEF tools tested elsewhere)	B,2, F.1
Electronic Notebook	B.1, A.2, C.1, F.1, F.2
Simulation Repository	A.1, D.1, D.2, E.1
Simulation Portal	D.3, E.4
Simulation Metadata	A.3., C2
Centralized Services	G.1

1.4. Test Strategy

The overall strategy for the NEESgrid acceptance testing is to verify that the NEESgrid components provided by the SI team meet the requirements defined in Section 1.2.
The SI will develop and document the formal test procedures and tools to be applied for each software module. These will be reviewed by the Consortium. The testing itself will be carried out by the SI and results submitted to the Consortium for review.

NEESgrid acceptance will consist of three types of tests:

- Unit/functional test for NEESgrid specific components. These tests are developed as part of the normal component development process. Unit tests are run throughout software development and prior to any software release. Unit tests validate compliance with all documented interfaces and also include stress testing, scalability testing, and testing of boundary cases where appropriate. Acceptance tests for NEESgrid include unit tests for components developed explicitly for NEESgrid (e.g., NSDS, NTCP, NMDS, NFMS). Components developed outside the scope of NEESgrid (e.g., NMI, CHEF) are represented in the Acceptance Testing Plan by workflow, gap, and packaging tests.
- Workflow/integration testing. These tests will cover specific system requirements described in Section 1.2, such as registration and delivery of streamed data, and search and download of data in a repository. In performing these tests, we will leverage current and planned experiments that are being performed in collaboration with earthquake engineering sites (MOST and other experiments, reflecting the distinct natures of structural, geotechnical, and tsunami experimentation). The acceptance testing protocol will require the SI to identify to the Consortium what integration tests will be covered as part of the experiment, and to provide for each integration test a brief summary as to the test status.
- Gap testing. Gap tests are equivalent to workflow/integration tests in all ways except for the context in which they are executed. Specifically, Gap tests are not conducted as parts of experiments performed in collaboration with earthquake engineering sites. (While Workflow/integration testing should cover most relevant usage scenarios, there will likely be integration tests that will not be covered under current or planned experiments. In these cases, the missing test cases will be documented via agreement with the SI and Consortium, and the specific tests conducted by the SI.)
- Packaging testing. Packaging tests cover issues related to the form in which NEESgrid components are delivered to the NEES Inc. Of particular importance are the capabilities provided for installing, and verifying installation of, software components.

Not all tests are simple pass/fail evaluations – some need to be performance related. That is, assuring that performance falls within acceptable ranges – established as part of the test procedures – will determine whether or not the component is acceptable.

Individual tests may be witnessed and test results will be reviewed and approved by NEES Consortium, Inc. as represented by the CONSORTIUM awardee and the IT Committee (a standing committee appointed by the Board of Directors).
The SI team will maintain records of all acceptance test results.

2. ACCEPTANCE PROCESS

2.1. Personnel

The Consortium's IT Committee will be charged with evaluating the acceptance test results and with final acceptance of both NEESgrid components and the overall NEESgrid system. It may elect to assign a subgroup of the committee membership for this task. However, final system acceptance shall be done by the entire committee.

2.2. Procedure

Acceptance Test results shall be evaluated using the following procedure:

- The SI provides the Consortium with detailed Acceptance Test Procedures and the Consortium accepts them.
- The SI alerts the Consortium of an upcoming test
- The Consortium's IT committee chair or his/her designee observes the test if and as desired
- The SI produces an Acceptance Test Report and submits it to the Consortium for review
- The Consortium formally reviews the Acceptance Test Report and produces a written review report indicating acceptance or non-acceptance.

2.3. Actions

Each functional component's Acceptance Test Procedure shall be performed by the necessary SI staff using the necessary resources. An Acceptance Test Report will be produced for each of the component acceptance tests. These results will be evaluated as described above.

2.3.1. Action upon Success

If upon evaluation the evaluation team finds that a functional component has successfully passed its acceptance test, that component shall be deemed accepted. The SI can then proceed with closure for this component, including final release and training as required.

2.3.2. Action Upon Failure

If after evaluation the acceptance team finds that a component has not passed its specified acceptance test, the SI shall be so notified and will be provided with a description of deficiencies. A checklist will be provided, based on the requirements agreed to by both the Consortium and the SI. The SI shall then proceed with further development and refinement and produce a revised version of that component. The revised version shall be retested and re-evaluated.

3. HIGH LEVEL TEST PLAN

3.1. Resources

Each individual acceptance test will require specific resources in terms of computer hardware, software, NEES equipment, etc. These required resources will be specified as part of each individual Acceptance Test Procedure.

3.2. Schedule

Individual Acceptance Test Procedures will be developed incrementally by the SI and reviewed/approved by the Consortium as soon as possible (with the final procedures to be defined no later than May, 2004). Tables 3, 4, and 5 indicate the dates associated with each incremental set of procedures .

Incremental testing of individual NEESgrid functional components will be done as soon as possible during the time period August 2003 through August 2004. Specific dates for each procedure will be proposed by the SI as part of the procedure definition activity.

Table 3. Developing Procedures for Unit Tests

Procedure for Component…	Due	SI Lead
NSDS Service	1/31/2004	Laura Pearlman
NSDS Client Libraries/APIs	1/31/2004	Laura Pearlman
NSDS Drivers/Plug-ins	1/31/2004	Paul Hubbard
NTCP Server and Java Client Library	10/15/2003	Laura Pearlman
NTCP C Client Library	12/15/2003	Lee Liming
NTCP C Gateway Plugin	12/15/2003	Lee Liming
NTCP Mplugin	12/15/2003	Lee Liming
NTCP Matlab Client Toolbox	12/15/2003	Erik Johnson
NTCP Matlab Backend Toolbox	12/15/2003	Erik Johnson
DAQ Code	09/30/2003	Paul Hubbard
DAQ Library APIs	09/30/2003	Paul Hubbard

Metadata Service	09/30/2003	Joe Futrelle
File Management Service	10/30/2003	Joe Futrelle
Repository Browser teamlet	11/30/2003	Joe Futrelle
Metadata ingestion tool	11/30/2003	Joe Futrelle
Central archiving service	01/30/2004	Joe Futrelle
Chef Base Distribution	11/30/2003	Chuck Severance
Chef Grid Services	11/30/2003	Chuck Severance
Chef MySQL	11/30/2003	Chuck Severance
Data Visualization Tools	11/30/2003	Chuck Severance
Telepresence Components (NTCP, NSDS and CHEF tools tested elsewhere)	11/30/2003	Nestor Zaluzec
Electronic Notebook	11/30/2003	Nestor Zaluzec
Simulation Portal Interfaces	3/31/2004	Greg Fenves
OpenSees API to Data Services	3/31/2004	Greg Fenves
Data/Metadata Repository Services	1/30/2004	Joe Futrelle

Table 4 lists the dates for specifying the procedures associated with workflow/ integration and gap tests. The SI team does not know at this time which of the workflow tests will be conducted as part of future large-scale collaborative experiments and which will be conducted separately. It will be specified as each test procedure document is completed, if possible, but may be deferred until closer to time that the test plan is executed.

Table 4. Developing Procedures for Workfllow/Integration and Gap Tests

Procedure for Function…	Due	SI Lead
Locate, retrieve, and view relevant experiment data and/or simulation code for a given research problem	12/15/2003	Joe Futrelle Chuck Severance
Locate NEESgrid participants with similar research interests	12/15/2003	Bill Spencer
Identify NEES facilities (availability and capabilities) relevant to a given experiment	12/15/2003	Doru Marcusiu Laura Pearlman
Chat with other NEESgrid users (text-only)	12/15/2003	Chuck Severance

Share documents and images with other NEESgrid users	12/15/2003	Nestor Zaluzec Chuck Severance
Set up and participate in a videoconference	12/15/2003	Erik Hofer
View numerical, video, and simulation data from published studies using CHE	12/15/2003	Chuck Severance
Select and view multiple channels (data and/or video) from experiment results side-by-side	12/15/2003	Chuck Severance
View multiple channels (data and/or video) with time synchronization (see esp. requirement E.3; also A.3 and B.4)	12/15/2003	Chuck Severance
Enter pre-experiment data into an electronic notebook (e.g., notes, equipment or sensor calibration data, video captures of specimen prep. process, etc.)	12/15/2003	Nestor Zaluzec
Enter preliminary or intermediate experiment results into an electronic notebook (e.g., data, still video captures, simulation model images)	12/15/2003	Nestor Zaluzec
View near-real-time data from a physical experiment	12/15/2003	Paul Hubbard Laura Pearlman
View near-real-time video from a physical experiment	12/15/2003	Nestor Zaluzec
Transport and store sensor data from a running experiment (see requirement B.2 for details)	12/15/2003	Paul Hubbard Laura Pearlman
Establish a client subscription to one or more live data streams	12/15/2003	Laura Pearlman
Produce a “virtual” data channel (server side data filtering/reduction/etc.)	12/15/2003	Paul Hubbard Laura Pearlman
Use a tool to excerpt information from an electronic notebook and use it to populate the metadata model	12/15/2003	Joe Futrelle Nestor Zaluzec
Use a free-text web form to capture user comments and insert them into the metadata model	12/15/2003	Joe Futrelle Chuck Severance
Use a tool to allow entry of structured metadata and populate the metadata model	12/15/2003	Joe Futrelle
Use a tool (or set of tools) to organize, classify, index, move, link, annotate, version, archive, and secure data in the data repository	12/15/2003	Joe Futrelle
Access and browse the contents of the simulation code repository	3/31/2004	Greg Fenves
Use the simulation portal to search for documentation, report bugs, submit new task proposals	3/31/2004	Greg Fenves
Perform simulations using the portal, storing models and results in the data repository	3/31/2004	Greg Fenves

Search simulation repository for models and results	3/31/2004	Greg Fenves
Customize the CHEF collaboration interface for a given user	12/15/2003	Chuck Severance
Use the following collaboration environment features via CHEF: scheduling, announcements, task management, information resource management, discussion, shared presentation, notification, email group management, email archive, and laboratory notebook	12/15/2003	Chuck Severance John Leasia
Use Grid security to authenticate (a user) to multiple resources in a single session via CHEF	12/15/2003	Laura Pearlman Chuck Severance
Use Grid information services to identify computation and storage resources via CHEF	12/15/2003	Sridhar Gullapali Chuck Severance
Integrate a web page and a web-based application into the CHEF environment	12/15/2003	Chuck Severance
Locate API and interface details describing how to integrate a complex application with the CHEF environment	12/15/2003	Chuck Severance
Remotely view an experiment conducted at one or more NEES equipment sites while collaborating with research partners to view the streaming data (from sensors and video sources) and quickly analyze the results in near real-time	12/15/2003	Sridhar Gullapalli Carl Kesselman
Review the entire design, specimen preparation, and test setup process for an experiment by accessing electronic notebooks	12/15/2003	Dan Abrams Bill Spencer
Run the same experiment at two distinct equipment sites using the same user interface tools (including especially data and metadata input and retrieval). (NOTE: The interfaces should be identical.)	12/15/2003	Sridhar Gullapalli Bill Spencer
Locate and view online training materials designed for engineers and scientists.	12/15/2003	Cristina Beldica

Testing procedures for the software packaging will be due in January of 2004, as shown in Table 5.

Table 5. Developing Procedures for Unit Tests

Procedures for …	Due	SI Lead
Software Packaging	01/15/2004	Doru Marcusiu

3.3. General Procedures

3.3.1. Unit Tests

Unit acceptance tests will be integrated with the regression testing procedures used during component development and software release. As described above, a testing procedure and testing report will be created for each component.

Unit tests will be automated, and to the extent possible the testing methodology used by the Globus Toolkit (summarized here) will be used when appropriate. Globus uses two different testing mechanisms: one for modules that are primarily C and one for modules that are primarily Java. These two mechanisms are unifed by the joint use of Tinderbox, which provides a common reporting interface for both mechanisms.

For our C modules the Globus team uses a home-made testing framework based on a Perl script known as "test-toolkit". The test-writer's interface is described at http://www.globus.org/gt2.4/test-toolkit.html. (This page was developed for use with iVDGL GLUE testing, which is why the specific tests it refers to are GLUE tests.)

Java module tests are based on the JUnit framework and can be run from within the component hosting environment. JUnit is a framework for developing unit tests for Java classes. JUnit framework provides a base class called TestCase that can be extended to create a series of tests for the class you are creating, an assertion library used for evaluating the results of individual tests, and several applications that run the tests. Tests can be run both on an hourly and on a daily basis depending on their criticality and how long it takes to run them. We have developed extensions to both the hosting environment and junit to facilitate Grid-specific testing. The **JUnit** extension allows us to test both collocated and remote servers easily without the test writers having to modify their tests. The **ant** extensions are used for stress testing, which allows the tests to run in multiple threads and from multiple JVMs without the test writers having to modify their tests. We also have a number of test conventions and patterns that allow us to easily run subsets of the tests (e.g., only run security tests or Open Grid Services

Infrastructure compatibility tests). Tinderbox is used as a centralized repository for all the test reports, but the presentation layer is not as good as it could be. Since all our test reports from **JUnit** are in HTML and XML we could easily customize better stylesheet conversions, which is something the SI plans to provide. See

http://choate.mcs.anl.gov:8080/tinderbox/OGSA/status.html

for a sample of Tinderbox output.

Test can be written using the Java **JUnit** interfaces and the API extensions we provide for stress tests and remote server tests. This testing environment requires a source download of GT3 core to run the test as well as the **junit.jar** binary from junit.org.

3.3.2. Workflow/Integration and Gap Tests

Integration/Workflow tests evaluate specific cross-component functionality with respect to realistic deployment environments. Integration/Workflow tests will include infrastructure services that are part of the NEESgrid system to be delivered to the Consortium (e.g., the centralized NEESgrid repository).

As described above, Integration/Workflow tests will be aligned to the extent possible with ongoing NEESgrid activity. Specific workflows to be tested will be documented, and a report will summarize the results of each test. Any significant usage modalities not covered under this process will be identified and specific test cases developed to address the omissions.

To date, two integration work flow efforts have debuted.

1) The Early Adopter Demonstration: Three equipment sites, the University of Nevada, Reno, Oregon State University and Rensselaer Polytechnic Institute were brought up to release 1.0/1/1 of the NEESgrid software.
2) The MOST Experiment: Two new Equipment Sites (UIUC) and U of Colorado are being brought up to pre Version 2.0 release of NEESgrid. We are debuting the Control Protocol, additional Collaborative tools and Data Management Services.

3.3.3. Packaging Tests

The SI deployment team will develop and execute the software packaging acceptance test plans. These test plans will ensure that the NEESgrid software suite can be installed, configured, and uninstalled on specified target systems in a reasonable manner.

The packaging test plan will largely involve performing the above actions with the software suite as released by the SI team on one or more target systems. The test plans will assume that all target systems meet the NEES-POP system requirements and other SI requirements as publicly specified.

After the packaging acceptance test plans are approved by the Consortium, the deployment team will notify the Consortium regarding when they may observe execution of the tests (should the Consortium wish to). All tests will be conducted by the deployment team and all results will be documented and archived for later use.