

SANDIA REPORT

SAND2014-20676

Unlimited Release

Printed November 3 2014

HyRAM Testing Strategy and Quality Design Elements

John T. Reynolds

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185-0748 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>



SAND2014-20676
Unlimited Release
Printed November 2014

HyRAM Testing Strategy and Quality Design Elements

John T. Reynolds
Mission Computing Services
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0748

Abstract

Strategy document and tentative schedule for testing of HyRAM, a software toolkit that integrates data and methods relevant to assessing the safety of hydrogen fueling and storage infrastructure. Because proposed and existing features in HyRAM that support testing are important factors in this discussion, relevant design considerations of HyRAM are also discussed. However, this document does not cover all of HyRAM design, nor is the full HyRAM software development schedule included.

ACKNOWLEDGMENTS

Dr. Katrina Groth, for direction and suggestions on schedule and for content suggestions.

Laura Connolly, for fitting this document into the corporate style framework.

CONTENTS

1	OVERVIEW	10
2	QUALITY DESIGN ELEMENTS	11
2.1	Persistent Unit Typing	11
2.2	A Plug-In Architecture.....	11
2.2.1	Wrapping Process	11
2.2.2	Physical Models	12
2.2.3	Interface Modules	12
2.2.4	Wrapper Scripts	12
2.3	Testing Strategies.....	12
2.3.1.	Unit Testing of Physical Models.....	12
2.3.2.	Model Input Range Verification	12
2.3.3.	Model Output Range Verification	12
2.3.4.	Scripted Testing	13
2.3.5.	User Testing	13
2.3.6.	Alpha and Beta Testing.....	13
2.3.7.	User Feedback.....	14
3	SCHEDULE.....	15
4	SUMMARY	16
4.1	User Communication and Feedback	16
4.1.1	Demos	16
4.2	Alpha Tester Contact List.....	16
4.3	Beta Tester Contact List.....	16
4.4	Bugzilla.....	16
4.5	Make Bugzilla Available to External Users.....	17
4.6	Roll out Bugzilla to External Users, Notify.....	17
4.7	Releases.....	17
4.8	Demos	17
4.9	Testing Features Still Outstanding.....	18
4.10	Schedule – Itemized.....	19
5	CONCLUSIONS.....	20

6 REFERENCES 21

7 DISTRIBUTION..... 22

TABLES

Table 4.1 Scopes 17
Table 4.2 Features 18
Table 4.3. Itemized Schedule 19

NOMENCLATURE

The following terms are used in this document.

Delegate: A delegate is a helper object given a task to perform by another section of code. This allows a task with steps that can vary widely due to differences in each task, to be treated identically across all of them.

Interface: In software development, this word is used in different contexts to mean different things. If used by itself, “software interface” should be assumed. HyRAM uses the word as defined below.

User Interface: The part of HyRAM the user can see and interact with. It includes forms, panels, screens, drop-downs, entry boxes, etc.

Interface Module: A piece of code supplied with a physical model, that makes all physical models look similar to HyRAM middleware. It reduces the level of expertise required by HyRAM developers to use the model, thus making the model easier to call.

Software Interface: A software interface is a piece of code designed to make multiple disparate tasks look the same to something that needs to interact with any of them in a consistent fashion. Software interfaces are required to interact with delegates, for example. An interface module is a form of software interface.

Physical Model: Provided by contributors well-versed in the nature of the physical process being modeled, and called by HyRAM.

Porting: This is the process of converting a set of instructions from one programming language, like MATLAB, to another, like Python.

Repurposing: This is the process of taking a suitable model that has already been created for use in another simulation or project and wrapping it in such a way that it can be called directly by HyRAM. Repurposing is distinguished from porting in that the original source code, or compiled binary, is packaged and called by HyRAM without the original source code being changed.

Session Database: HyRAM stores all input data and selected output data in an in-memory database called a Session Database. This database is queried by modules to retrieve user input from separate screens and to populate UI input elements with default or previously-entered data. Session data is saved and loaded as single document, and can be saved and loaded by the user as needed.

Standard Output, STDOUT: This is the file descriptor or stream to which text is written while a console application is running. Text written directly to the screen when running a console program is an example of standard output.

Wrapper Script: This is a script automatically generated by HyRAM middleware to enable, in a consistent fashion, the calling of a physical model interface script with data supplied by the user.

1 OVERVIEW

The HyRAM design includes a number of features that will enable testing on multiple levels.

Parts of the design related to software quality and testing are discussed in this document.

The user interface and middleware are written in C#. This language is well-suited to the task because it has strong typing, object orientation and a rich set of generic data structures, along with built-in functionality to enable creation of a framework capable of combining contributions from a large community with a disparate skillset into an extensible, maintainable and usable software package.

Contributors knowledgeable about the physical processes being modeled are more likely to be conversant in MATLAB¹ and Python, than C#. It is expected that algorithms might be contributed in any language, but Python is the only language supported in the current version. HyRAM is built in such a way that contributions can be plugged in without disrupting the existing design. It is important to the HyRAM software development team that it be possible to validate software simulating physical processes separately from the user interface. HyRAM's design enables diversely-sourced software to be incorporated and called without requiring modification and recompilation, either by the core team or end user, and either from the user interface or an automated process.

¹ HyRAM currently does not call MATLAB code, although a module could be plugged in to enable that without design changes. But it does allow Python modules, and it is easier to port code from MATLAB to Python than from MATLAB to other languages.

2 QUALITY DESIGN ELEMENTS

HyRAM middleware contains design elements to improve robust operation and to enable stability and maintainability, such as

2.1 Persistent Unit Typing

HyRAM enforces an immutable link between values and the units that define them. Input values are stored in SI units, and the only way to retrieve or set those values is by specifying compatible units at the time data is submitted to or requested from the session database. Conversions are performed implicitly by the system. So the user can select units s/he is comfortable with in the user interface, while values sent to a physical model are always of the correct and expected unit. At lower levels simple unit conversion is enabled by specifying a conversion factor. A conversion delegate is created and applied for more complicated conversion. Explicit unit conversion is never required by the developer or user, however.

2.2 A Plug-In Architecture

Models are plugged into HyRAM by writing or repurposing an existing model, writing an interface module to simplify wrapper creation, and generating a wrapper script to call its software interface. That process is described in this section.

Although any language can be used for model creation or repurposing², currently HyRAM supports only Python. HyRAM avoids any sophisticated language features in its model binding process, instead choosing to call console applications or scripts that accept arguments on the command line and write output to the standard output stream. This will eventually enable the end user to add models to the system and control them from the user interface without having to understand a complicated linking methodology, and enables the host application (HyRAM) to communicate without having to understand details about the programming language being used.

New or preexisting physical models are added to HyRAM by the creation of interface modules on the plug-in side and wrapper scripts on the HyRAM side. This combination of interface modules and wrapper scripts enables the use of complicated and powerful physical modules in a simple and consistent fashion.

2.2.1 Wrapping Process

At run time, HyRAM extracts source code or binaries from storage that represent the physical process being modeled and generates a wrapper script. It then calls the wrapper script, waits for it to complete, and collects results. Finally, it deletes all of the executable code and data collected during the plug-in and execution process, and returns selected data to the program and/or user.

² If an implemented model exists that fits a need within HyRAM, it can be repurposed to fit into HyRAM's framework by writing a software interface script. This prevents the need to create that model from scratch.

2.2.2 Physical Models

Physical Models are provided by contributors well-versed in the nature of the physical processes HyRAM is modeling. These models may be written from scratch specifically for inclusion in HyRAM, or they may be requested from the original authors and repurposed.

2.2.3 Interface Modules

Interface modules are generally created by the plug-in contributor. Their purpose is to present a simple procedural interface to HyRAM middleware. This allows HyRAM to call model code that may be complex and require a high level of domain knowledge to operate, in a simple, consistent fashion.

2.2.4 Wrapper Scripts

HyRAM generates Wrapper Scripts based on information it is given when the wrapper delegate object is created.

2.3 Testing Strategies

HyRAM testing strategies include unit testing, scripted user-interface manipulation and user testing. Short- and medium-term implementation of these strategies is discussed below.

2.3.1. Unit Testing of Physical Models

There was significant attention given to creating an infrastructure that can support the ability to plug in physical models. As a result, calling these models, whether from the user interface or an automated test script, is relatively simple. The calling process makes a simple function call wherever the model is needed. The same mechanism that allows plug-ins can be leveraged to allow the creation of a list of models, along with inputs and expected outputs, to be tested by an automated process. This process will call each model in turn and examine its outputs for correctness. It can then generate a report that becomes part of the verification process for each release. Unit testing can also be leveraged while models are being called in a normal HyRAM user session. It will be possible to toggle this feature as required.

No unit testing facility has yet been implemented.

2.3.2. Model Input Range Verification

Each model wrapping process must include a range of valid values for each argument passed to a model. Arguments should be tested for validity before allowing HyRAM to call the model, or at least to let the user know that the model is being used outside of approved input ranges. Because each model wrapper is inherited from a base class, this functionality will be added to that class on a timeframe defined in the schedule shown later in this document, and will become available to all models simultaneously.

2.3.3. Model Output Range Verification

It becomes more difficult to examine outputs for correctness as the number of models added to the system increases. There are multiple ways to examine output and the more of these that can be applied, the higher the likelihood of discovering problems before users do and the more this process can be automated. These tactics are presented for model output validation, by increasing level of complexity:

- 1) Define a range of valid values for each output and examine each value to determine whether it's in the accepted range. It is fairly easy to write software to perform this task, and this software can be used to dramatically reduce the number of things requiring human examination. Value ranges can be adjusted to reduce or increase the amount of output requiring expert examination.
- 2) Define relationships between multiple outputs and acceptable values in those relationships. This task can be non-trivial. Delegates can be created to examine output. But this scenario would also require a consistent method for retrieving the associated rules from the physical model contributor, and for storage. This may present design issues. This would likely require the submission of a document (perhaps comments in the interface module), that describes the relationships and values, with each contributed interface module.
- 3) Define relationships between inputs and acceptable output values, separate from or in concert with the previous item. This would probably be appropriate in a limited set of circumstances, and would require additional model contributor input. Similar to the previous item, this task would be offloaded to delegates. The same delegate interface can be used (and possibly should, for simplicity), but if used would require additional input.

Model Output Range Verification can be used in these cases:

- 1) With an automated process that calls all models in turn and generates a model quality report;
- 2) Throughout normal HyRAM operation. For performance reasons, this behavior can be toggled on and off as needed.

2.3.4. Scripted Testing

A printable user test script must be created and maintained. This script can be used to guarantee that every program feature is exercised and evaluated before release. Each member of the development team is primarily focused on the tasks s/he is performing. Scripted testing makes sure that all parts of the software get attention periodically, and help prevent the accumulation of undocumented defects. Scripted testing should be performed and documented periodically and on a scheduled basis.

2.3.5. User Testing

This schedule includes distributing HyRAM to a selected group of industry and laboratory professionals that, as broadly as possible, will represent the expected HyRAM user base. Distribution schedule is tentative due to its dependence upon approval by Sandia's Intellectual Property (IP) Department. Members of the test community should be added any time appropriate, but especially when new features are added to HyRAM that need the perspective or expertise of members not found in the current alpha test community.

2.3.6. Alpha and Beta Testing

Software is distributed to Alpha testers when the feature set is not yet complete and may contain known defects. When the feature set is largely complete and most issues are deemed solved,

software is released to a larger community of expected users who are likely to exercise the software in myriad unexpected ways and report defects previously unknown and more difficult to uncover. This document may expand in the future to include Beta testers.

2.3.7. User Feedback

The user test community isn't just expected to report on software defects. We intend to iterate on features and look-and-feel as well. Getting the software into the hands of the users is the best way to find out what works and doesn't, about look-and-feel issues and about features that would make HyRAM useful. Part of the compensation expected from users will be feedback.

These tools will be used to solicit feedback from users:

2.3.7.1 Bugzilla

A Bugzilla site at <https://h2zilla.sandia.gov> is used for

- Defect reports
 - Problems experienced with the software
 - Incorrect output
 - Questionable output
- Feature requests
 - Look-and-feel issues
 - Lack of current distributed model library to meet user needs
 - Model interaction assumptions that make HyRAM difficult to use for user scenarios

2.3.7.2 User Surveys

Formal user surveys will be periodically created and sent to users. The completion of these surveys will be a part of the user's obligation as licensee.

3 SCHEDULE

This schedule includes

- 1) Software releases at least monthly
- 2) Software features not yet available and needed to realize testing goals enumerated in this plan
- 3) Demos
- 4) User reviews
- 5) Feedback solicitation

4 SUMMARY

The spreadsheet below contains a list of milestones. These tasks are included:

4.1 User Communication and Feedback

4.1.1 Demos

Demonstrations of existing and new features will be provided periodically. These will serve the following purposes:

- 1) Showcase existing and new capabilities
- 2) Teach users how to access capability provided
- 3) Allow user feedback and approval on features and look-and-feel
- 4) Allow feedback from users about their experience with HyRAM in a public forum where users can interact with the development team and each other.

4.2 Alpha Tester Contact List

A sampling of users from across the industry must be compiled. The list should be diverse but not too large. It should contain between 10 and 50 users. Too few, and there won't be enough feedback to represent usage of a larger user base. Too many, and we could be inundated with feedback and unable to be as responsive to issues as we would like.

There are two tasks in this category:

- 1) Compile – Get and vet a list of names
- 2) Contact, Request Participation – Contact the users identified. Make it clear that they must respond to participate, and what their obligations are.
 - a. Continue iterating on the list until the optimal number of users, with the correct diversity of use, have been identified and have agreed to participate in the Alpha test program.

4.3 Beta Tester Contact List

When HyRAM has a feature set that is nearly complete and Alpha testers have demonstrated a high degree of confidence in its use, a list of Beta Tester contacts should be compiled. The initial number of users should be capped at 100. Compilation and contact details are the same as for the Alpha test community, and all users in that community should be automatically added to this list. When feedback from the initial Beta test community has reached a manageable level, community size can be expanded or HyRAM should be officially released, depending on the level of quality the software has obtained.

4.4 Bugzilla

Bugzilla is used to request features and report defects. The URL is <https://h2zilla.sandia.gov>. The site is live, but is still protected behind Sandia's firewall.

4.5 Make Bugzilla Available to External Users

There are a few issues to iron out before Bugzilla can be made visible to users outside of Sandia. These issues will be resolved soon. Having the site visible is not the same as giving everyone access. An informal vetting process will be used and it is expected that a few principals outside of Sandia will be notified on an as-needed basis.

4.6 Roll out Bugzilla to External Users, Notify

Concurrent with the first Alpha release, the Bugzilla site will be rolled out to external users. As users are added to the Alpha (and Beta) rolls, they will be given access to Bugzilla.

4.7 Releases

Very frequently, due to important bugfixes or functionality changes, HyRAM will be released. There are two scopes:

Table 4.1 Scopes

Scope	Frequency and Availability
Internal	Bi-weekly. Immediately available to the entire development team inside Sandia. User community will not be notified.
External	Approximately every two months and at any time deemed appropriate by the Principal Investigator. They will be placed on a download site and the user community will be notified.

4.8 Demos

Software demos will be performed at conferences, when appropriate visitors are hosted at Sandia, and at any time deemed appropriate by the Principal Investigator.

4.9 Testing Features Still Outstanding

Some HyRAM features have not yet been designed and implemented. These are not necessary for a testing and release plan, but *are* necessary for the level of quality expected by this team:

Table 4.2 Features

Feature	Description
Unit Testing Middleware	Design and code necessary to allow physical models to be tested independent of the user interface.
Automated Unit Testing Middleware	Design and code necessary to enable testing of physical models to be automated by script.
Automated Unit Testing User Interface	Design, code and user interface elements necessary to create a test script without writing code, or to launch a predefined process to generates tests and reports results.
Realtime Unit Testing Facility	Design, code and user interface elements necessary to automatically check input and results of physical models while the application is being used to perform real work. The user will be able to turn this behavior on and off by a simple command in the user interface.
Test Script	A document stored in Subversion that a designated tester will be expected to follow, verbatim, in order to fully exercise all of the functionality presented in HyRAM. To be effective, it should be kept updated as the feature set changes.

4.10 Schedule – Itemized

Table 4.3. Itemized Schedule

This schedule is expected to change frequently over the course of this project effort. In the final version of this document, this schedule will serve as a future schedule management metrics source.

Done?	Due Date	Task	Deliverable	Lead	Notes
Yes	11/10/2014	2.1 - HyRAM	HyRAM Demo - ISO TC197 WG24	Groth	
No	11/11/2014	2.1 - HyRAM	Bugzilla use cheatsheet	Parkins	Send to Alice and Chris for their feedback
No	11/14/2014	2.1 - HyRAM	Make Bugzilla available to external users	Malashev	
No	11/14/2014	2.1 - HyRAM	Compile alpha tester list	Groth	
No	11/14/2014	2.1 - HyRAM	Input Range Verification Middleware Design and implementation	Reynolds	
No	11/16/2014	2.1 - HyRAM	Contact alpha testers, request participation	Groth	Target is 20-40 users. Users who respond are more likely to participate
No	11/28/2014	2.1 - HyRAM	Audit alpha testers list, add/notify new users as appropriate	Groth	Target is 20-40 users
No	12/17/2014	2.1 - HyRAM	Create user test script	Parkins	
No	12/18/2014	2.1 - HyRAM	Perform scripted test and start writing tutorial based on 'indoor fueling' test case and report	TBD	KG will ask Sarah Jane
No	12/19/2014	2.1 - HyRAM	Review/update user test script	Parkins	
No	2/25/2015	2.1 - HyRAM	Perform scripted test	TBD	Non-developer
No	2/27/2015	2.1 - HyRAM	Deliver first HyRAM Alpha Release	Groth	Dependent upon approval by IP
No	2/27/2015	2.1 - HyRAM	Roll out Bugzilla to external users; notify	Reynolds	
No	3/1/2015	2.1 - HyRAM	Design Unit Testing facility middleware	Reynolds	See Unit Testing of Physical Models section
No	3/3/2015	2.1 - HyRAM	Implement Unit Testing Facility Middleware	Reynolds	
No	3/5/2015	2.1 - HyRAM	Design Automated Unit Testing facility middleware	Reynolds	
No	3/9/2015	2.1 - HyRAM	Implement Automated Unit Testing facility middleware	Reynolds	
No	TBD	2.1 - HyRAM	HyRAM Demo	Groth	
No	4/4/2015	2.1 - HyRAM	Implement Automated Unit Testing facility UI	Parkins	Will include specific code for each Python model
No	TBD	2.1 - HyRAM	HyRAM Demo	Groth	
No	4/14/2015	2.1 - HyRAM	Perform scripted test	TBD	Non-developer
No	TBD	2.1 - HyRAM	HyRAM Demo	Groth	
No	3/10/2015	2.1 - HyRAM	Implement "toggleable" realtime unit testing (UI)	Parkins	See Scripted Testing section

5 CONCLUSIONS

This document was created to assist the HyRAM team and stakeholders in understanding what was considered when putting together a testing plan for HyRAM, to enhance understanding of the HyRAM design and software features impacting quality and to give the HyRAM team and stakeholders a shared understanding about testing, quality assurance and related featureset schedule for this product.

Every successful software development schedule has some fluidity due to changing needs, manpower and requirements. This document is also fluid. Please contact the author, or HyRAM Principal Investigator Dr. Katrina Groth, to be sure that you have the latest version.

6 REFERENCES

7 DISTRIBUTION

1	MS0748	John T. Reynolds	Org. 9341
1	MS0748	Katrina Groth	Org. 6231
1	MS0899	Technical Library	9536 (electronic copy)

