NASA Data Acquisition System Software Development for Rocket Propulsion Test Facilities

Phillip W. Hebert, Sr.[1] and Alex C. Elliot[2]
NASA, Stennis Space Center, MS, 39529

Andrew R. Graves[3]
Camgian Microsystems, Stennis Space Center, MS, 39529

ABSTRACT

Current NASA propulsion test facilities include Stennis Space Center in Mississippi, Marshall Space Flight Center in Alabama, Plum Brook Station in Ohio, and White Sands Test Facility in New Mexico. Within and across these centers, a diverse set of data acquisition systems exist with different hardware and software platforms. The NASA Data Acquisition System (NDAS) is a software suite designed to operate and control many critical aspects of rocket engine testing. The software suite combines real-time data visualization, data recording to a variety formats, short-term and long-term acquisition system calibration capabilities, test stand configuration control, and a variety of data post-processing capabilities. Additionally, data stream conversion functions exist to translate test facility data streams to and from downstream systems, including engine customer systems. The primary design goals for NDAS are flexibility, extensibility, and modularity. Providing a common user interface for a variety of hardware platforms helps drive consistency and error reduction during testing. In addition, with an understanding that test facilities have different requirements and setups, the software is designed to be modular. One engine program may require real-time displays and data recording; others may require more complex data stream conversion, measurement filtering, or test stand configuration management. The NDAS suite allows test facilities to choose which components to use based on their specific needs. The NDAS code is primarily written in LabVIEW™[4], a graphical, data-flow driven language. Although LabVIEW™ is a general-purpose programming language; large-scale software development in the language is relatively rare compared to more commonly used languages. The NDAS software suite also makes extensive use of a new, advanced development framework called the Actor Framework. The Actor Framework provides a level of code reuse and extensibility that has previously been difficult to achieve using LabVIEW™. The NDAS team also uses the agile development cycle for all of the development activities.

---

[1] Lead, Software Engineering Group, Engineering & Test Directorate, EA-34, Stennis Space Center, MS, 39529; AIAA Senior Member.
[2] Computer Engineer, Software Engineering Group, Engineering & Test Directorate, EA-34, Stennis Space Center, MS, 39529; non-AIAA member.
[3] Software Engineer, Camgian Microsystems, Test Operations Contractor, B-8306, Stennis Space Center, MS, 39529; non-AIAA member.
[4] LabVIEW™ is a trademark of National Instruments (NI). Neither NASA Stennis Space Center, nor any software programs or other goods or services offered by NASA Stennis Space Center, are affiliated with, endorsed by, or sponsored by National Instruments.

I. Introduction

There were two primary drivers for developing the NDAS software suite. The first was NASA's recent resumption of operations at Stennis Space Center's large test facilities after thirty years of contractor control. With the advent of the commercial space launch industry and the evolving potential for testing a wider variety of contractor engines, the need for a non-proprietary data acquisition system to support government and commercial engine testing became a priority. The software provides the government with unlimited rights and guarantees privacy of data to commercial entities.

Second, driving data acquisition system consistency within Stennis test facilities and to adjacent propulsion test facilities could provide cost savings to NASA. Test stand engineers, test conductors, and other stakeholders could be transitioned to different test facilities with shorter learning curves to operate data acquisition systems. In addition, with an increasing user base, efficiencies of scale could be utilized to provide a better software product with a wider variety of features to all users.

II. Purpose

Stennis Space Center's primary mission is to perform testing of chemical rocket propulsion test articles and provide test data to test article customers. Data acquisition systems are utilized to acquire and provide this data.

A test facility may control the test article (i.e. rocket engine) or a test article controller may control the test article. The test article customer determines the specific implementation of test article control. Figure 1 shows an example of the facility controlling the test article and a data acquisition system interface to the test article. Figure 2 shows an example of a test article controlled by an engine controller, with a vehicle simulator, and a data acquisition system interface to the test article.

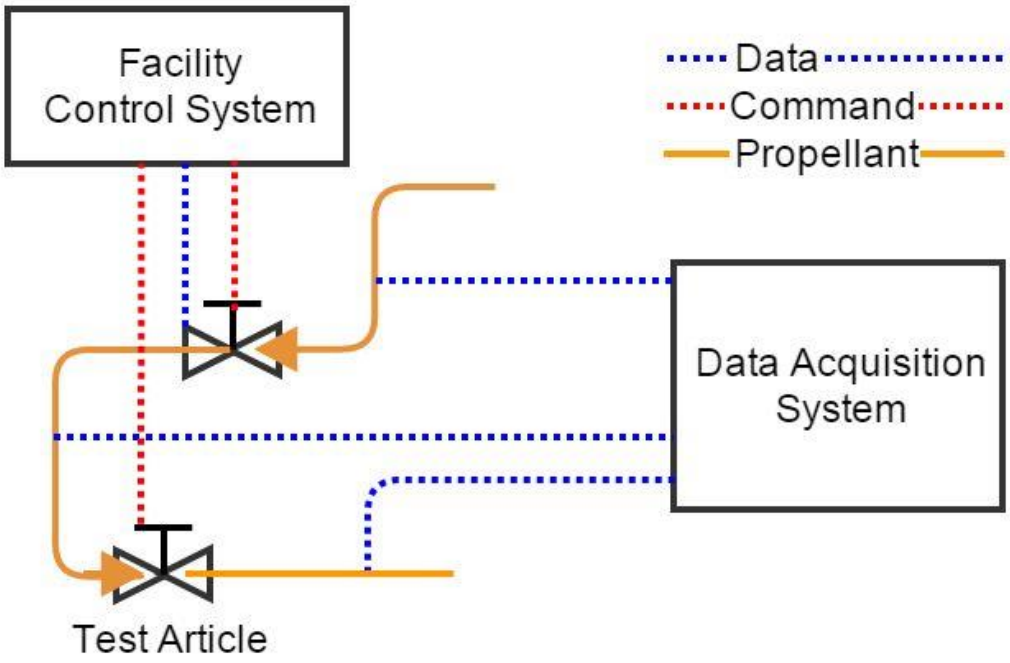

Figure 1. Acquisition with facility controlled test article

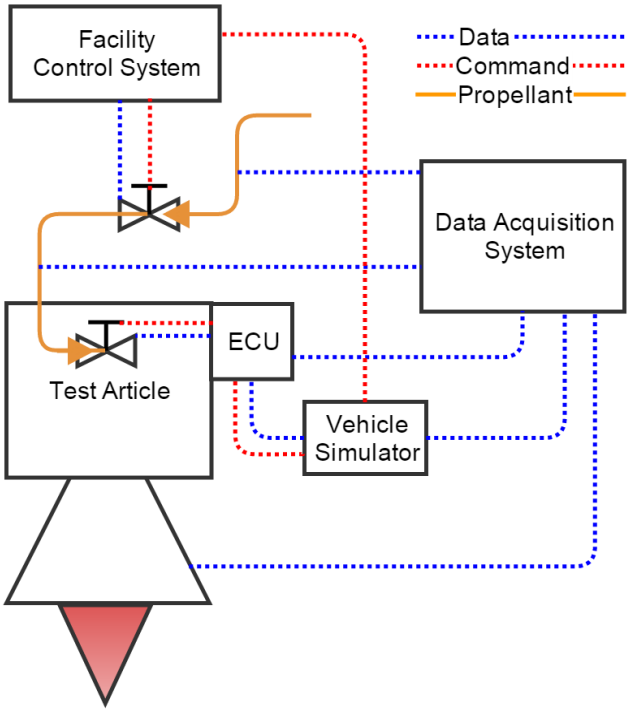

Figure 2. Acquisition with engine controller and simulator

Data acquisition systems require software to perform the required functions of a data acquisition system, including acquiring, storing, and distributing data. The NASA Data Acquisition System (NDAS) is a software suite designed to operate and control many critical aspects of rocket engine testing. The software suite combines real-time data visualization, data recording to a variety formats, short-term and long-term acquisition system calibration capabilities, test stand configuration control, and a variety of data post-processing capabilities. Additionally, data stream conversion functions exist to translate test facility data streams to and from downstream systems, including engine customer systems.

III. Requirements Development

The first step in the engineering process was to gather and document a set of requirements for the software. Representatives from all rocket propulsion testing centers participated in the requirements gathering effort. During the fall of 2010 this effort included visits to three of the four RPT centers: Stennis Space Center, White Sands Test Facility, and Plum Brook Station. The resulting requirements set captured much of the functionality of the existing proprietary software set as well as operational and capability enhancements intended to improve the efficiency of operation of the software. The original requirements set included provisions for future work to operate and control the data acquisition system hardware for all four propulsion test centers.

IV. Initial System Development and Implementation

The need for such a system was most prevalent at Stennis Space Center's A-Complex and B-Complex test facilities. Therefore, the project decided to implement first prototype at the A2 Test Facility during testing of the J-2X rocket engine. Due to the need to stand up a functional system quickly, the decision was made to develop the software with limited capabilities, based upon a sub-set of the software requirements. The software would run in parallel with a proprietary software set on a secondary DAS at this facility. At the time, the project was named NASA DAS or NDAS. The initial limited capability set was called Phase I of the project and the subsequent phase to implement the remainder of the requirements was called Phase II. The team chosen to implement the NDAS software was based at Stennis Space Center and included NASA civil servants and on-site contractors.

V. NDAS System Overview

The primary design goals for NDAS are flexibility, extensibility, and modularity. Providing a common user interface for a variety of hardware platforms helps drive consistency and error reduction during testing. In addition, with an understanding that test facilities have different requirements and setups, the software is designed to be modular. One engine program may require real-time displays and data recording; others may require more complex data stream conversion, measurement filtering, or test stand configuration management. The NDAS suite allows test facilities to choose which components to use based on their specific needs.

NDAS incorporates a hardware abstraction layer called the NDAS Translation Layer (NXLT) and NDAS Calibration Translation Layer (NCXLT), allowing easier adaptation of the software to different hardware platforms. In addition to the hardware abstraction layers or hardware translation modules, the architecture groups the software into modules based on function. The NDAS modules include the main operation and control module (NOPS), the engineering unit processing module (NPRO), the display module (NDIS), the calibration module (NCAL), the data logging module (NLOG), the data stream gateway (NGATE), and the instrumentation and roadmap database (NIRD). The one stop shop (NOSS) serves as the front-end for all user interaction with NIRD. Figure 3 depicts the high-level NDAS architecture.[5]

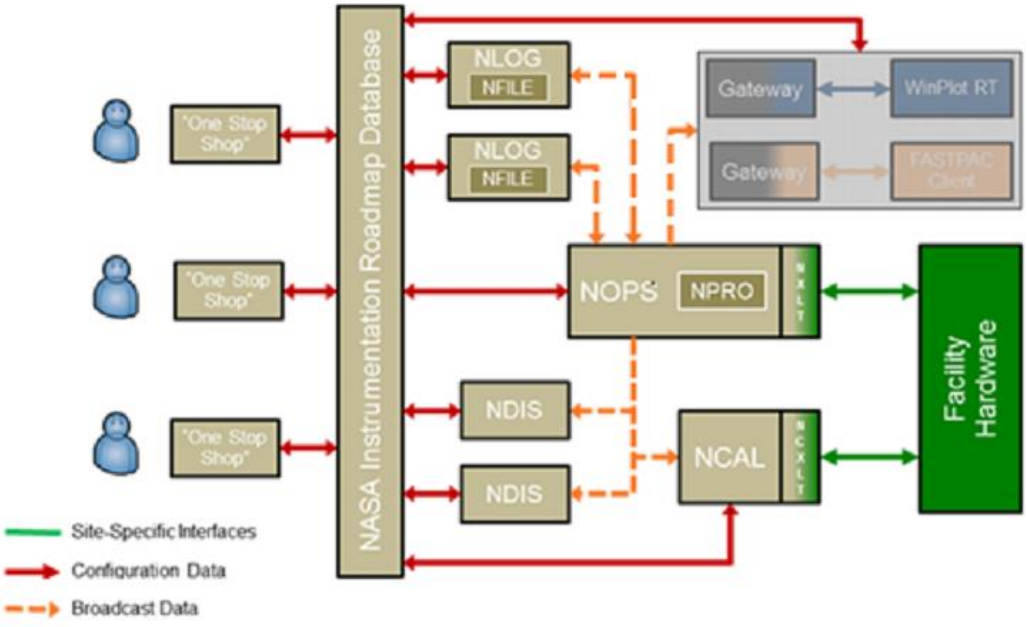

Figure 3. NDAS Functional Overview[6]

A. NOPS

NOPS is the central command and control software that manages the data acquisition hardware, streaming of data to NDAS applications, and provides the framework in which NPRO executes the run-time engineering scaling and user defined calculations. NOPS is comprised of a collection of interconnected processes and user interfaces coordinated by a central controller process. Through the user interface, the user may load and validate a system configuration, start and stop acquisition and data streaming, and view status and client connection information. The

[5] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 12.
[6] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 12.

background processes provide hardware abstraction, run-time data processing, and streaming of data to networked clients.[7]

Below (Figure 4) is a screen capture of the main NOPS display.

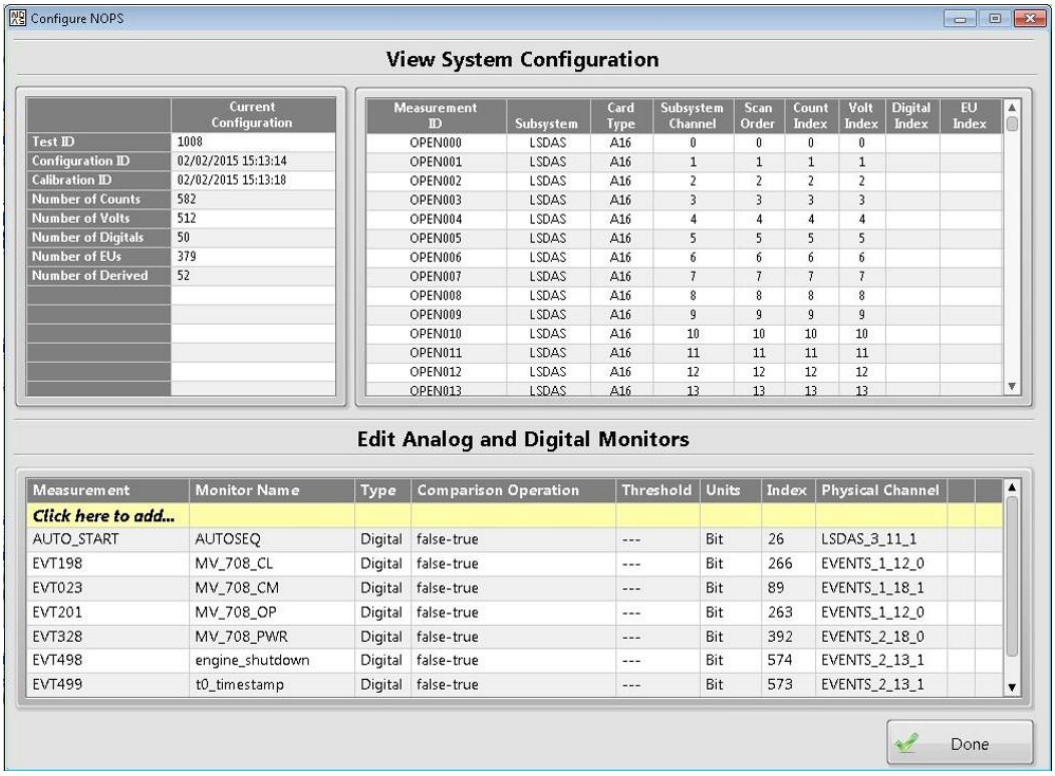

Figure 4. Main NOPS Display

1. NXLT

The NXLT module is the translation layer of the software. This module communicates directly with the site-specific software. This software also communicates with all of the DAS hardware and can send commands to the hardware as well as receive data from the system.

2. NPRO

The NPRO module contains the functionality for converting the data to engineering units for storage and display. NPRO provides engineering unit (EU) data back to NOPS. The raw data is read from the time slice (i.e. the NOPS data stream) and the EU data is written back into the time slice. Raw and EU data can have different indexes into the data stream (i.e. processed data can be placed into a different array location in the scan list than the raw data).

NPRO runs as an application programming interface (API) to both NOPS and NLOG. It does not modify the time slice; it reads volts or counts and writes into the EU array. EU conversions read volts from the time slice and convert volts to EUs.

NPRO receives all parameters required for EU conversion from NIRD. This includes but is not limited to measurement type, first order calibration coefficients created by NCAL, higher-level polynomials, resistive temperature detectors (RTDs), laboratory calibration coefficients, referenced measurements for thermocouples and barometers, and user's defined formulas for derived measurements. Commonly used measurements are hardcoded to speed execution. NPRO has the capability to accept formulas for customer-specific measurements as well. As new

[7] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 13.

calculations are introduced, the quantity and utility of the measurement are evaluated to determine if the calculation should be hard-coded in NPRO or remain as a derived measurement.[8]

B. NCAL

The NCAL application contains the functionality that monitors the health of the data acquisition system and verifies the integrity of the data system in meeting data uncertainty requirements. NCAL provides system calibration capabilities, measurement system analysis (MSA), and the calculation of errors due to linearity and hysteresis.

The architecture allows for flexibility in defining a calibration process, which may differ among centers or test programs. NCAL interfaces with hardware through the NCXLT translation layer. All access to the DAS hardware is achieved through this high-level NCXLT interface.

NCAL acquires data through the NOPS data stream, reads the NOPS network stream server, records and analyzes calibration result data, is self-contained with no reliance on separate downstream loggers/processors, interfaces directly with the database through the NIRD API, and retrieves and commits calibration routines and results with full audit control.

The NCAL software is object-oriented utilizing the LabVIEW™ object-oriented programming (LVOOP) feature of LabVIEW™ to improve scalability and maintainability. The calibration process itself is based upon an object-oriented concept where the overall general system calibration is comprised of a series of more specific and self-contained calibration steps, belonging to one of several "classes" or "types".

NCAL is designed to allow users to construct and execute a calibration procedure according to their specific requirements. A calibration procedure is the top-level execution call. The procedure is made of one or more calibration instructions. Each calibration instruction is a specific calibration type that executes a configurable sequence of commands over a series of eight distinct steps. A calibration instruction is used to calibration a group of one or more measurements.

The design of NCAL allows users to combine different types of calibrations in any order to customize the system wide calibration process. NCAL user interfaces allow the user to control the execution of system calibrations and display results. NCAL also provides an interface for manual control of the signal conditioners and calibration equipment.

The NCAL design is based around two state machines. The main state machine handles the user interface interactions and the other controls the calibration steps using the information in the instruction. The instruction provides the various commands each calibration type is configured to run at each step.

All NCAL procedures and instructions are stored in the NDAS database.

[8] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 26.

Below (Figure 5) is a screen capture of the main NCAL display.

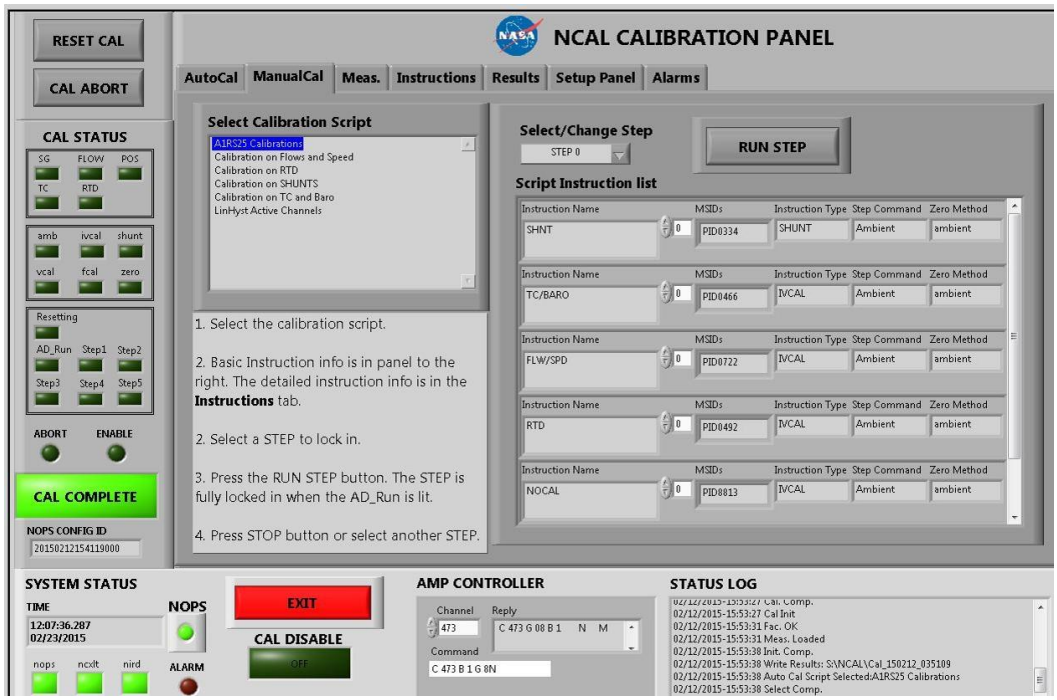

Figure 5. Main NCAL Display

C. NLOG

NLOG is the data logger module that reads data from the NOPS data stream and metadata from the NIRD database. It records the data and metadata as groups, in a technical data management streaming (TDMS) file format. The logger can record data as a continuous file or in a first-in-first-out (FIFO) buffer directory.

The logger's file converter can also read in a TDMS file and convert them to different formats. Optionally, the original file size can be reduced by selecting fewer input data groups and/or selecting the start and stop sample numbers.

The output file formats for NLOG are TDMS, WinPlot[9], MATLAB®[10], comma separated values (CSV), and Hierarchical Data Format 5 (HDF5). During file conversion, the raw data can be reprocessed with new coefficients downloaded from the NIRD database.

Deselecting unwanted groups can also reduce the non-TDMS output types in size (example: deselect counts and volts group to record only the EU group of channels), and/or select a start and stop sample to convert when converting from TDMS.

Unless otherwise requested, all output data files are recorded in a standard directory, using a standard naming convention.[11]

[9] WinPlot is a NASA developed product used to view time-domain data.
[10] MATLAB is a registered trademark of The MathWorks, Inc.
[11] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 55.

Below (Figure 6) is a screen capture of the main NLOG display.

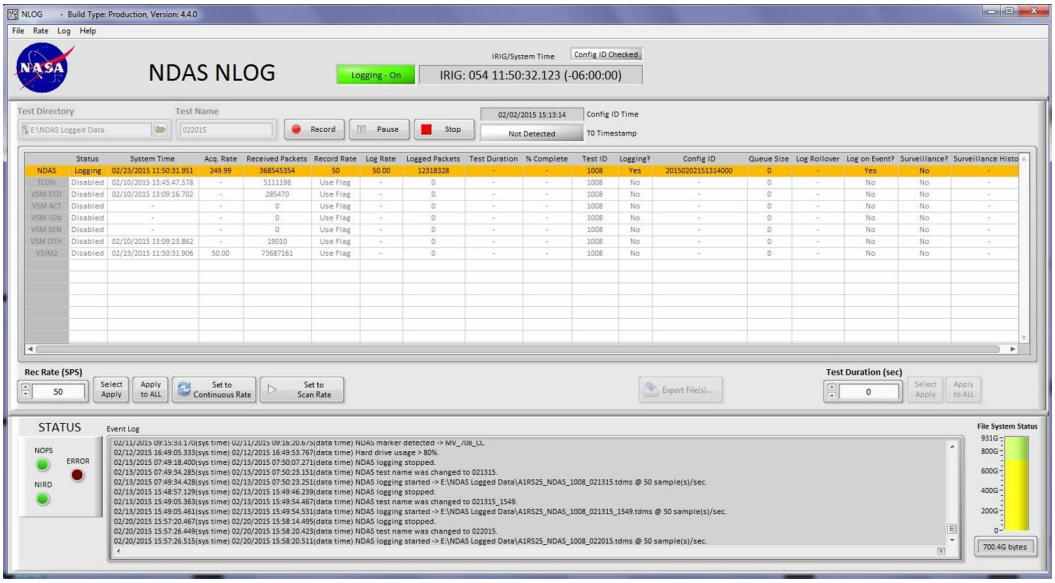

Figure 6. Main NLOG Display

D. NDIS

The NDIS module is the real-time data user interface. It provides data display capabilities including both graphical and tabular displays. The NDIS functionality is accessible from various workstations on the data acquisition network.

The NDIS application communicates amongst multiple parallel processes using a combination of producer consumer loops, queued state machines, and a factory pattern. The code is built using LabVIEW™ objects; some elements may not be immediately recognized, but the same underlying principles apply.[12]

Below (Figure 7) is a screen capture of the main NDIS display.

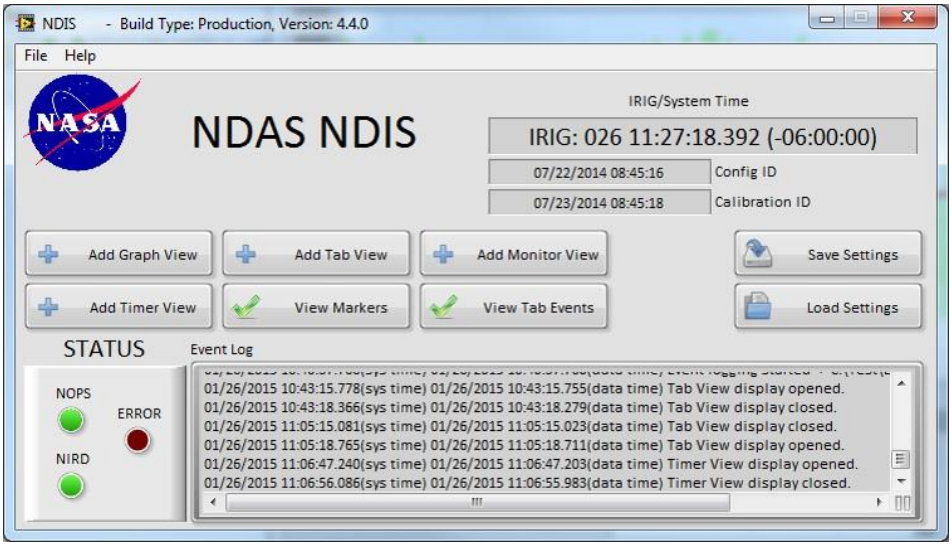

Figure 7. Main NDIS Display

[12] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 81.

E. NGATE

NGATE provides the conversion of the NDAS native real time data stream format to the customer specific real time data stream format. Figure 8 below depicts an example of how NGATE interfaces with NDAS and test facility systems.

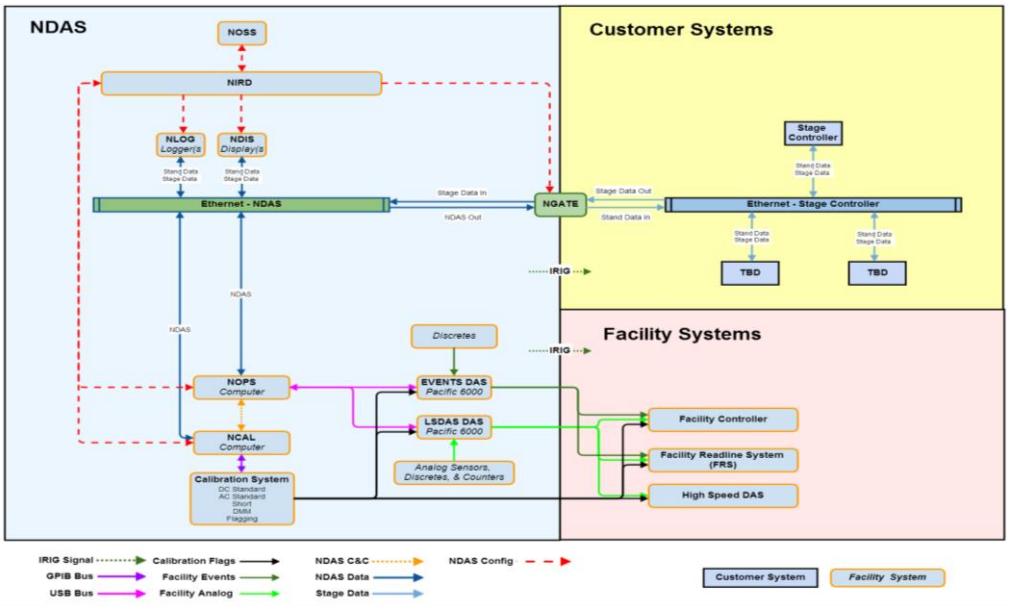

Figure 8. NGATE Component within NDAS

Below (Figure 9) is a screen capture of the main NGATE display.

Figure 9. Main NGATE Display

F. NIRD

The NIRD database module provides the storage function for the test facility instrumentation system configuration, including measurement devices and the routing of cabling to the DAS input hardware, DAS hardware

settings such as gain and filter settings, and acquisition rates. The roadmap is a subset of the database that details the configuration for a specific test. The use of a database allows linkage of data elements and the creation of dependencies. The roadmap serves as the governing document for the test configuration of the DAS.

G. TDMS File Viewer

The TDMS File Viewer allows for review and editing of the native TDMS file format. It provides functionality to view and modify metadata as well as perform common file tasks. Although WinPlot is the standard tool for plotting data, the TDMS File Viewer is capable of simple plotting and tabular data display.[13]

H. Calibration Editor

The Calibration Editor is an application that provides NDAS calibration file viewing, editing, and report generating functions. The calibration editor is based on an event driven state machine that executes predefine tasks on user selected calibration files. In addition to viewing and editing capabilities, it allows the user to commit and retrieve previously stored results from the NIRD database. The Calibration Editor has a set of configuration parameters that must be setup for the program to function properly.[14]

VI. NDAS Technology

While primarily written in LabVIEW™, a graphical, data-flow driven language where program execution follows 'data flow' along wires, NDAS employs a multitude of technologies as depicted in Figure 10. NDAS runs on the Windows 7®[15] operating system. NOPS, NDIS, NCAL, NGATE, and NLOG are written in LabVIEW™. The underlying NIRD database is defined using MySQL™[16]. NOSS utilizes a multitude of technologies, including a web browser, Hyper-Text Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript®[17], Apache™[18], and Hypertext Preprocessor (PHP™)[19].

A. LabVIEW™

LabVIEW™ was originally chosen as the primary development tool for NDAS for several reasons:

- It excels at hardware/software interfaces,
- It has extensive libraries of hardware drivers for virtually all lab equipment and data acquisition systems,
- It has extensive libraries for data manipulation and analysis,
- It allows for rapid development of a rich user experience (UX) with theme-able UI controls and indicators, and
- It has an extremely large and enthusiastic user/developer community.

---

[13] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 141.

[14] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014, page 148.

[15] Windows 7® is a trademark of the Microsoft Corporation. Neither NASA Stennis Space Center, nor any software programs or other goods or services offered by NASA Stennis Space Center, are affiliated with, endorsed by, or sponsored by the Microsoft Corporation.

[16] MySQL™ is a trademark of Oracle Corporation. Neither NASA Stennis Space Center, nor any software programs or other goods or services offered by NASA Stennis Space Center, are affiliated with, endorsed by, or sponsored by Oracle Corporation.

[17] JavaScript® is a trademark of Oracle Corporation. Neither NASA Stennis Space Center, nor any software programs or other goods or services offered by NASA Stennis Space Center, are affiliated with, endorsed by, or sponsored by Oracle Corporation.

[18] Apache, Apache Foo, and Foo are trademarks of The Apache Software Foundation. Used with permission. No endorsement by The Apache Software Foundation is implied by the use of these marks.

[19] PHP is a trademark held by PHP.net.

NDAS has leveraged some advances in the LabVIEW™ language and available application frameworks to build a mature, flexible, and extensible suite of applications.

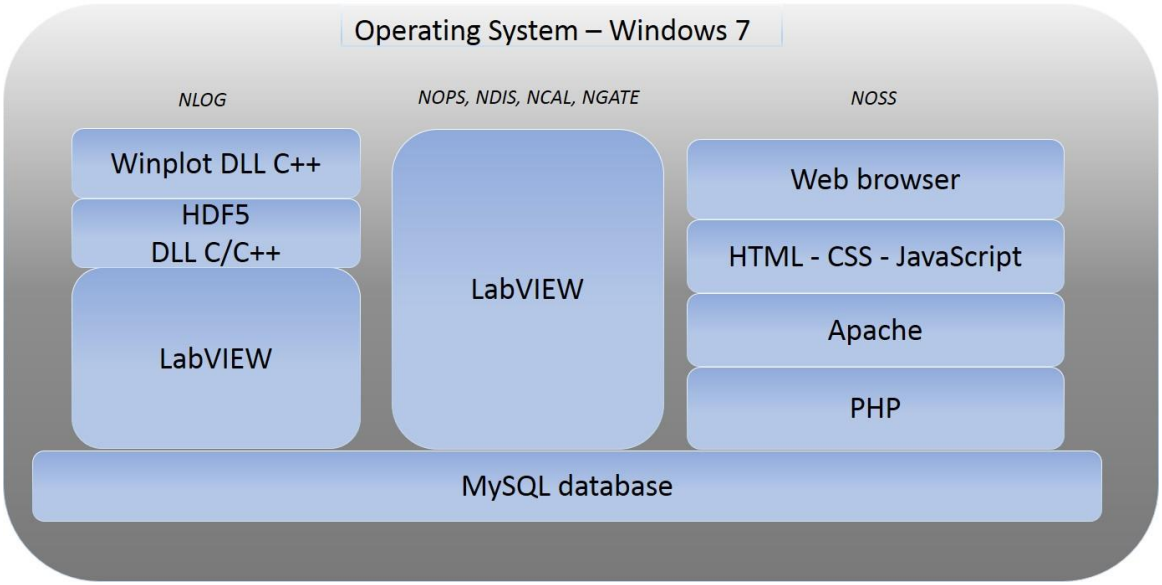

Figure 10. NDAS Technology Stack

B. Actor Framework

In order to achieve the design goals of maximization of code reuse and extensibility, the NDAS software suite makes extensive use of a new, advanced development framework called the Actor Framework. The Actor Framework leverages the command pattern and a queued message handler to provide developers with standard actor class for implementing message-based parallel processing. Developers spend time extending the actor behavior with methods and messages instead of troubleshooting the starting, stopping, and management of parallel processes.

All LabVIEW™ developers extend the same actor class delivered with the LabVIEW™ development environment, allowing code sharing within a team and within the global LabVIEW™ community. The Actor Framework is the primary framework for each application in the NDAS suite. This framework implements a model-view-controller (MVC) architecture, allowing models and controllers to be used across multiple applications with only changes to the presentation layer.

This may not seem revolutionary but prior to the Actor Framework, doing something as simple as a MVC architecture required a “home-grown” solution that was time consuming to implement and maintain.

Figure 11 shows the simplification in code using the actor framework.

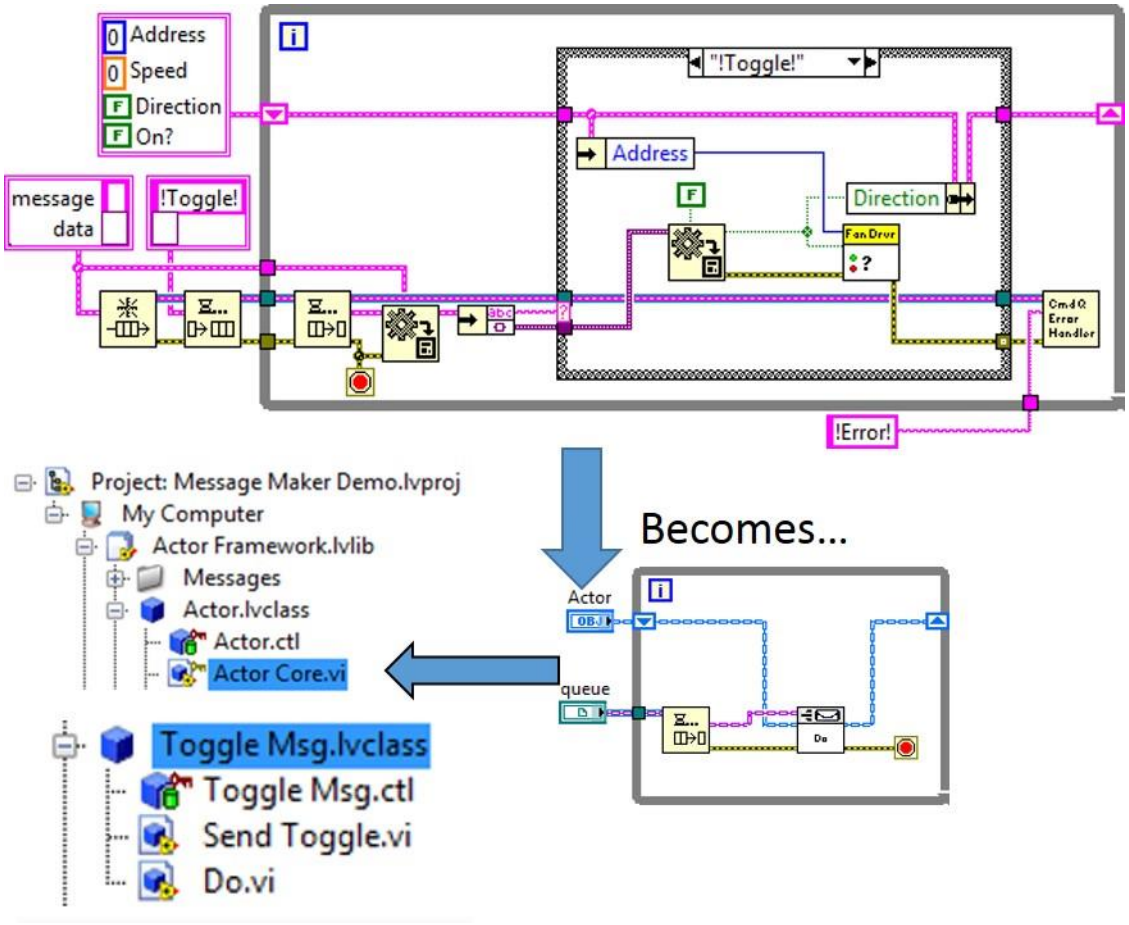

Figure 11. Actor Framework[20]

C. NDAS Hardware Abstraction Layer using Actor Framework

The NDAS Hardware Abstraction Layer (HAL) uses the Actor Framework as shown in Figure 12. The abstract actor defines the API for the hardware abstraction layer. A parallel process is responsible for managing the high-level state of the acquisition hardware and providing a stream of data to the NOPS application as the data is read from the hardware. This implementation provides a consistent API regardless of the specific hardware acquisition hardware and serves as a wrapper around low-level hardware drivers that are loaded at run-time based on the system configuration.

[20] Images from "Introduction to the Actor Framework", presented by Stephen Mercer and Allen Smith of NI during NI Week 2011.

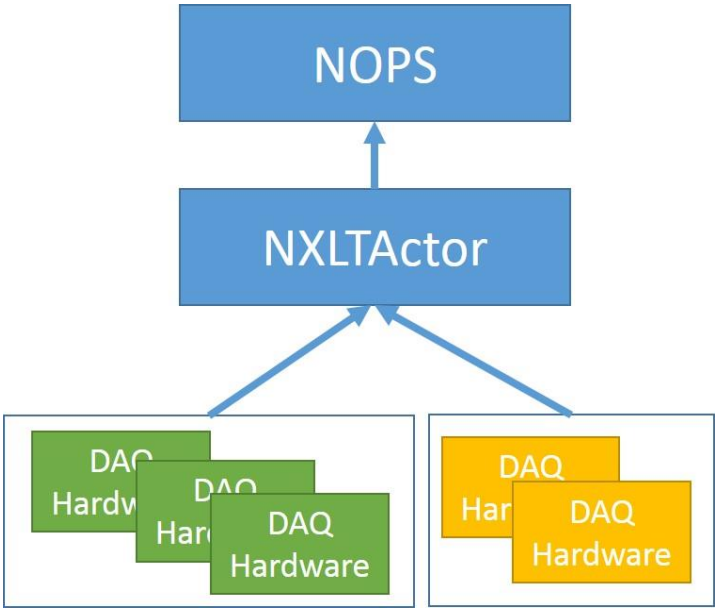

Figure 12. NDAS Hardware Abstraction Layer using Actor Framework

SSC data acquisition hardware currently implements two different architectures. One architecture integrates the signal conditioning required for interfacing with the instruments required to perform the test with the analog-to-digital converters that convert an analog signal into a digital signal, which can then be used by computers. The other architecture segregates these two functions. The HAL provides transparency between the data acquisition hardware and the NOPS module. Figure 13 shows how the HAL, using the actor framework provides this transparency.

1. *Scenario 1 – Integrated data acquisition (DAQ) and signal conditioning*

The analog-to-digital conversion, scaling, and filtering (signal conditioning) functions are performed within a single system, which is typical of most modern data acquisition systems. The NXLT actor converts high-level API messages directly to hardware commands.

2. *Scenario 2 – Independent Digitizer and Signal Conditioners*

The analog-to-digital conversion, scaling, and filtering (signal conditioning) functions are split between two or more pieces of hardware, which is typical of legacy NASA systems. The NXLT actor converts high-level API commands into one or more commands and routes them to the appropriate driver.

Thus, from the perspective of the NOPS module both scenarios are identical. This software architecture allows for the portability of the software to many data acquisition hardware platforms, maximizing the reuse of code.

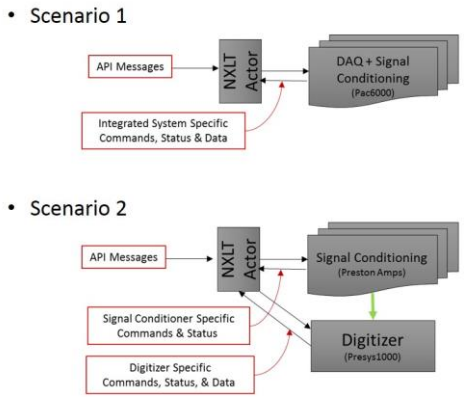

Figure 13. NDAS Hardware Abstraction Layer Scenarios

VII. Software Life Cycle and Development Methodology

A software life-cycle model provides a framework for the planning of the overall technical effort and implementation of the technical processes. This framework includes life-cycle phases, milestone decision gates, major technical reviews, event entry and success criteria, and other key intermediate events leading to project completion. In order to accomplish delivery of the product required for an aggressive project schedule, the NDAS project selected the agile life cycle.

The agile life cycle incorporates a development methodology based upon iterative and incremental development. After the initial release of the software requirements, the software development process is segmented into iterations. Each iteration consists of requirements analysis and refinement, design, implementation, and testing. At the end of each iteration, a set of code for review is completed.

Advantages of the selected methodology emphasize developing software and providing functionality instead of comprehensive documentation, involvement of the customer early and often, and rapid response to changing environments and priorities while being able to continuously refactor and improve the product.

Prior to the development of the NDAS software suite, there have been no larger scale software development efforts at Stennis Space Center. As such, all stakeholders involved in software product development were required to adjust rapidly to large-scale software engineering processes. Part of the maturity process involved making organizational changes, software development method changes, adapting to evolving user requirements, and developing industry standard source code management and deployment as the project progressed.

A. Agile Scrum Software Development Process

Figure 14 below depicts the agile scrum life cycle used for software development at SSC.

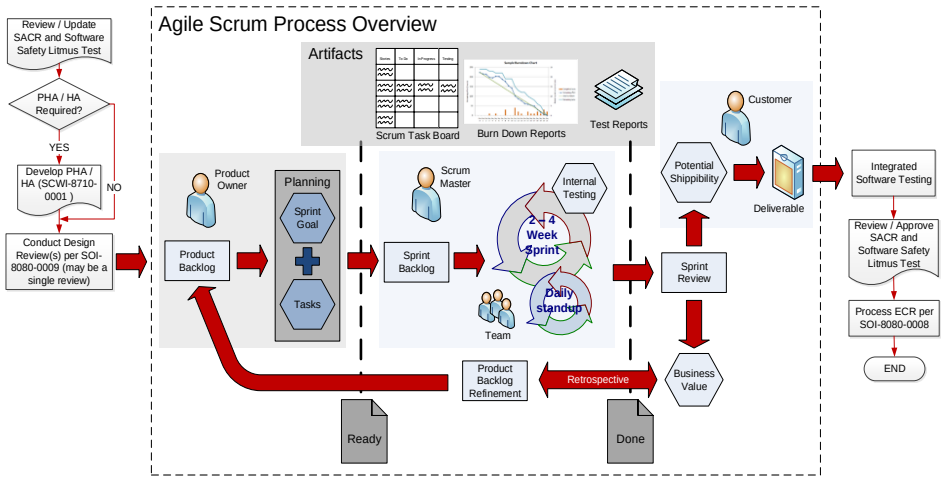

Figure 14. Agile Scrum Software Process

The agile scrum software process used segments the development effort into "sprints" each lasting two-weeks, in which additional functionality is incorporated into the product during each sprint as well as correction of any problems discovered in the field with use of NDAS. The team, along with input from stakeholders, determines the priority of functionality to include in each sprint.

Each day the team conducts a stand-up meeting to address activities performed from the previous day, planned activities for the upcoming day, and any impediments in which a team member may be experiencing.

At the end of each sprint, the team conducts a retrospective meeting to review the tasks completed in the previous sprint, demonstrate completed functionality, address what went well during the previous sprint, and address ways to improve future sprints. Tasks unable to be completed during the previous sprint are assigned to the product backlog to be completed in the next sprint.

Functionality completed at the end of a sprint is delivered for testing in a laboratory environment prior to being delivered for use at the test facility. Once laboratory testing is complete, the software is delivered for final testing in the field environment at the test facility. Once field-testing is complete, the software is submitted for formal release using SSC's software configuration management process.

Figure 15. Software Development Rhythm depicts an example of the agile life cycle process.

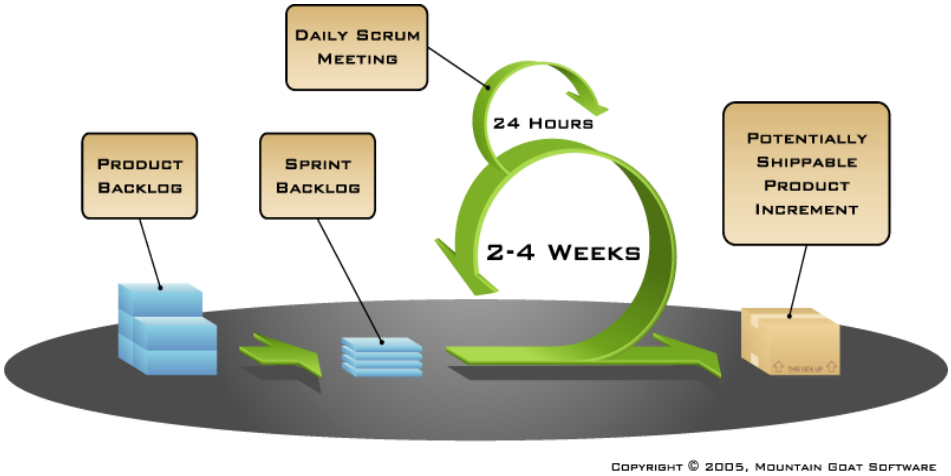

Figure 15. Software Development Rhythm

B. Source Code Management

The NDAS project uses software development, collaboration, and tracking tools from Atlassian®. These tools include JIRA®, Confluence®, and Stash™. JIRA® is an issue tracking project management tool. Confluence® is a collaboration tool. Stash™ is a software repository management tool.

Because NDAS is a multi-developer team, proper management of the source code is critical to the success of the project. Figure 16 depicts how source code management within the NDAS project.

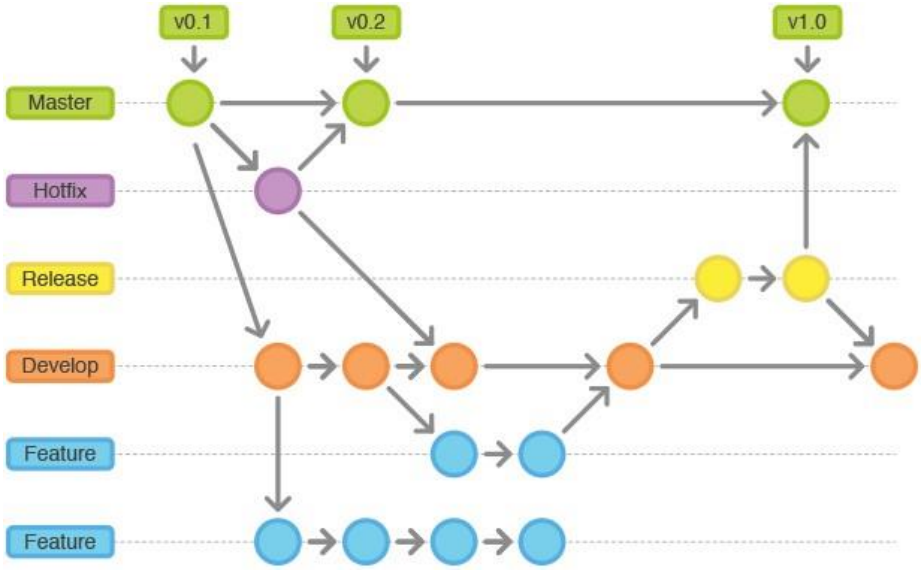

Figure 16. Source Code Management

C. Software Development Workflow

Figure 17 depicts the workflow used for NDAS software development. Tickets are created using JIRA® and assigned to individual team members. Once a team member begins working on a ticket, its status is marked as "in progress" until complete. Once complete, they are submitted for a "pull request" to be included in the next build to be delivered to testing. Once QA has passed, the ticket is then closed.

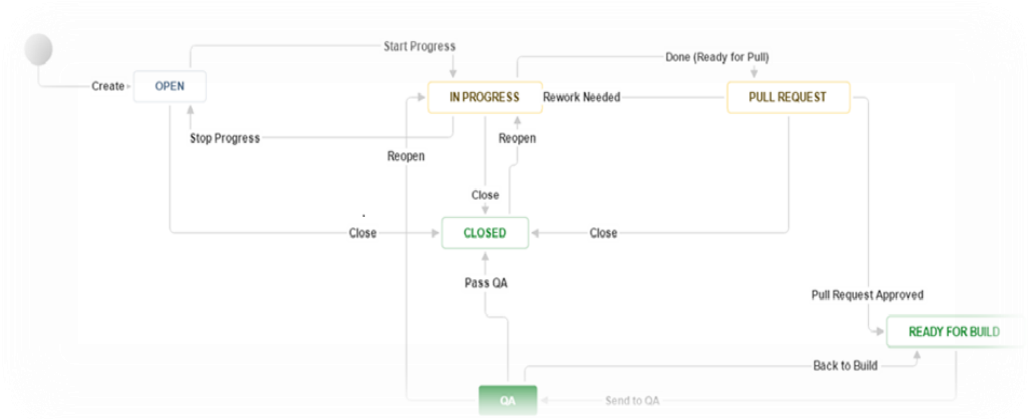

Figure 17. Software Development Workflow

VIII. Summary

The NDAS software was designed and developed from scratch at Stennis Space Center. The team uses widely available software development tools such as Atlassian JIRA®, Stash™, and Confluence® to manage software development, the LabVIEW™ Integrated Development Environment (IDE) for code development, and MySQL™ databases to store data and configurations.

The NDAS software has a broad range of capabilities including; data acquisition and distribution, web-based system configuration interface, real-time graphing and configurable data displays, data logging and file conversion to multiple formats, automated calibrations, long-term system calibrations, engineering unit conversions, real-time derived calculations with measurement inputs, and definition of measurement redlines and data markers.

The original goal for the software was to develop a common data acquisition software suite for NASA propulsion test centers, however, the software could one day be transferred to commercial organizations and other government agencies involved in any data acquisition endeavor. The NDAS software suite is best suited for large scale, multiuser data acquisition systems with lower-speed (250 samples per second or below) acquisition requirements.

Stennis Space Center is the largest of the four NASA propulsion test facilities including Marshall Space Flight Center, Plum Brook Station, and the White Sands Test Facility. There are currently eight test stands at SSC, which require data acquisition systems, and these other centers add even more potential uses. Launch facilities such as Kennedy Space Center have ground systems for data acquisition and control as well. Outside of NASA, the defense industry operates additional test facilities in the rocket engine testing domain. In the future, the NDAS software could be deployed for wider use at these facilities.

References

Proceedings

[1] Images from "Introduction to the Actor Framework", presented by Stephen Mercer and Allen Smith of NI during NI Week 2011, Charts 4, 16, 22.

Manuals

[2] NASA Data Acquisition System (NDAS) Software Developer's Manual UG-000012, Revision: 3.0, May 4, 2014.