Implementing the Magdalena Ridge Observatory Interferometer Supervisory System

Allen Farris[a], Robert Blasi[a], Robert Kelly[a], Louis Jencka[a], John Young[b], and Eugene Seneta[b]

[a]New Mexico Institute of Mining and Technology, Socorro, New Mexico, USA
[b]Cavendish Laboratory, University of Cambridge, Cambridge, UK

ABSTRACT

The Magdalena Ridge Observatory Interferometer (MROI) software system contains distributed systems managed by a centralized Supervisory System. Interface software is generated from spreadsheets that describe commands, monitor points, and fault conditions for each subsystem. The Supervisory System consists of an Executive, Operator, Database Manager; one or more Supervisors plus Fault Manager, and Data Collectors.

System-wide simulations are discussed: (1) a test framework is generated from the spreadsheets characterizing a subsystem; (2) a detailed simulation of the actual hardware in a subsystem; (3) a system-wide simulation of collecting astronomical data based on executing observing projects. The first two levels have been implemented.

Keywords: software design, control system, simulation, distributed system, interface protocol

1. MROI PROJECT

The Magdalena Ridge Observatory Interferometer (MROI) is a federally and institutionally funded facility being built and managed by the New Mexico Institute of Mining and Technology. It is located west of Socorro, New Mexico at an altitude of 10,500 feet. The interferometer will accompany a fast-tracking 2.4 m telescope, currently in operation. Many details of the general design of the interferometer, its specific sub-systems and its scientific objectives have been discussed at previous SPIE meetings on optical interferometry. Reference 1 is a discussion of overall design of the system and Reference 2 gives a status update.

The MROI is designed to support 10 1.4 meter unit telescopes in an equilateral "Y" configuration, operating in the visible and the near-infrared. Each unit telescope operates on an altitude-altitude mount utilizing only a primary, secondary, and tertiary mirror to inject starlight into the fast tip-tilt system and then into the beam transport system. These unit telescopes can be relocated onto a subset of 28 stations. The first telescope is now complete and is being tested on site.

The key science mission for the MROI is centered on three main areas: (1) studies of the environments of Active Galactic Nuclei (AGN); (2) stellar formation and the earliest phases of planet formation; and, (3) fundamental physics: stellar diameters, mass-loss, mass-transfer, convection and pulsation of single and multiple star systems.

Major systems of MROI include: (1) Supervisory System; (2) Unit Telescope, including the mount, optics, enclosure, wide field acquisition, and fast tip-tilt systems; (3) Beam Relay System; (4) Vacuum System; (5) Delay Line System; (6) Fringe Tracker; (7) Beam Combiner System; (8) Automated Alignment System; and (9) Environmental Monitoring System.

The MROI project is broken into a number of work packages that are being designed and fabricated independently. A significant fraction are being fabricated, and in some cases designed, by third parties. Our collaborators at the University of Cambridge function as overall system architects and are also responsible for implementing several of the key systems, including the delay lines and fast tip-tilt system.

The MROI beam combining facility was completed in February 2008. It includes all interconnected buildings on Magdalena Ridge housing the control facilities, optical, electrical and mechanical laboratories, delay lines, vacuum system, beam combiners and detectors.

Further author information: (Send correspondence to Allen Farris)
Allen Farris: E-mail: afarris@mro.nmt.edu

2. MROI SOFTWARE SYSTEM

The MROI software system is a collection of hierarchically structured distributed systems, corresponding to the major functional divisions of the interferometer (Unit Telescope, Delay Line System, Automated Alignment System, etc.) managed by a centralized Supervisory System. Generic interface software, in Java or C, is automatically generated from spreadsheets that describe system-specific commands, monitor points, and fault conditions for each subsystem. This software communicates over gigabit Ethernet using a TCP/IP protocol that is independent of the subsystem implementation language. This code generation framework also generates a standalone test environment for each subsystem. The spreadsheet is also used to describe data to be monitored, collected, and stored in a centralized database. The subsystem is executable as a completely operational, standalone system. The architecture of the MROI software system has been presented at previous SPIE meetings. Reference 3 gives an overview of its design. This paper will present a current view of its functioning. The Supervisory System and the generic interface software has been completely designed and its basic functionality implemented. Detailed testing is ongoing and enhancements are planned, especially in the Operator, Supervisor, and Fault Manager components.

With 10 unit telescopes, the MROI system will contain a network of over 100 devices, mostly computers, with more than 100 software applications running on those computers. The Supervisory System must manage all devices and applications to form a coherent structure that is capable of doing useful work. An interferometer is orders of magnitude more complex than the software that runs a single dish telescope. The complexity comes from having to manage many resources over a distributed network, not from the number of telescopes.

In such a system there are three major problem areas: (1) developing an approach for managing resources and integrating applications into an overall execution framework that executes projects and preserves data being collected; (2) developing a strategy for cold starting all computers and software applications, including initialization and bringing them to an operational state; and (3) developing a strategy for gracefully stopping all software applications to preserve all data being produced. To solve these problems, the Supervisory System communicates with all applications over the network using gigabit optical fiber, ethernet and the TCP/IP protocol.

The MROI software system must merge independently developed systems into a coherent, robust, and unified software system that is intended to last for more than twenty years. We have deliberately not placed unreasonable constraints on applications, allowing independent development and testing within a software environment based on compatibility with TCP/IP, Linux, C, Java, and Xenomai (a real-time add-on to Linux). This aspect of MROI is significant because it allows the flexibility needed for future system expansion; over its lifetime we expect technology improvements in many areas, hardware as well as software. A major strategy in developing applications is that they are not required to implement a client/server protocol. Rather, properties of the application are specified using spreadsheets that contain most of the information in a formal Interface Control Document (ICD). These spreadsheets are input to a code generation framework that automatically generates the necessary layer of client/server software. This process is the basic mechanism by which an application is integrated into the MROI software system framework.

3. MROI SUPERVISORY SYSTEM

The MROI Supervisory System consists of the following components: an Executive, an Operator, a Database Manager, one or more Supervisor plus Fault Manager, and one or more Data Collectors. The Executive starts, stops, and monitors the status of all active systems, including components of the Supervisory System. The Operator component is the mechanism by which the telescope operator interacts with the Supervisory Systems. The Database Manager manages all access to the permanent database, including accessing and updating configuration data. The Supervisor and Fault Manager are tightly coupled and each Supervisor has its own Fault Manager. The Supervisor manages one or more unit telescopes by issuing commands to subsystems to support scientific observations, system calibrations, or diagnostic testing. The Fault Manager determines the significance of all faults and takes appropriate action to handle them; such action is highly dependent on the operational state of the array. The Data Collectors collect engineering monitor data and science data from all active systems, reformatting data to be permanently stored in the database.

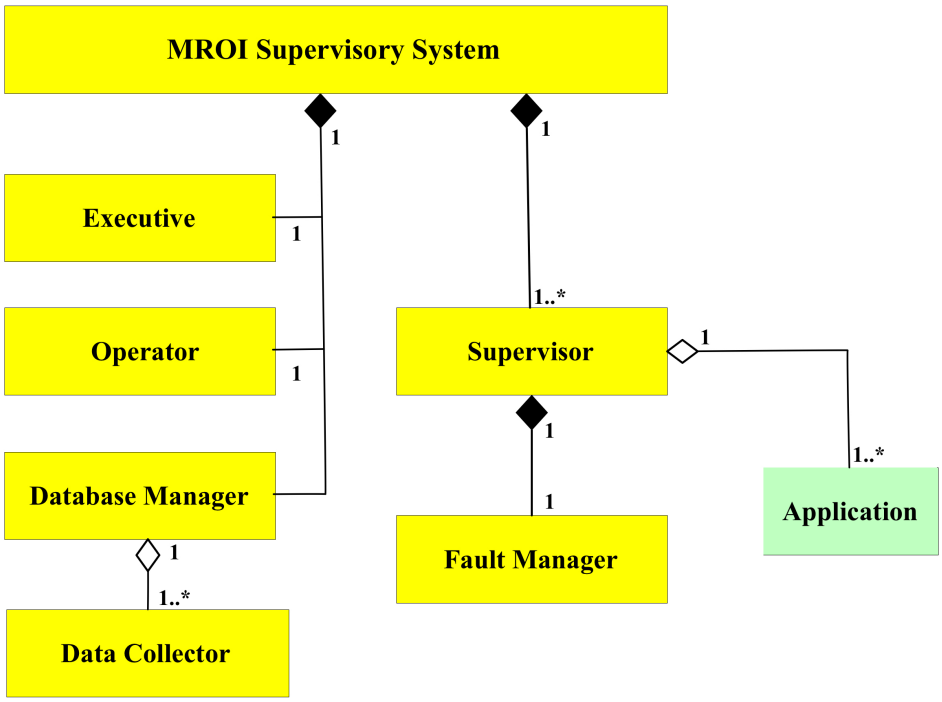

Figure 1. **Structure of the MROI Supervisory System**

The Supervisory System, depicted in Figure 1, manages the entire MRO Interferometer. It is responsible for starting, initializing, monitoring, and stopping the entire system, as well as executing commands to do useful work. A centralized database system is integral to this software system. A special startup program accesses the database to retrieve a pre-defined startup scenario and starts the Database Manager, Executive and Operator. It also deploys Data Collectors that collect all engineering monitor data and science data from all active systems, reformatting data to be stored in the database. Data Collector instances collect data from different application systems and may run on computers anywhere in the system. There is only one Operator interface, even though this module may interact with more than one operator workstation.

When the special startup program brings the Database Manager, Executive and Operator to an operational state, the Executive then takes over, and starts the entire system, including all application systems and the remaining portions of the Supervisory System itself. The Executive uses this pre-defined startup scenario to determine which systems to start and their configuration. For example, the Executive may start more than one Supervisor. A Supervisor manages one or more unit telescopes to support scientific observations, system calibrations, or diagnostic testing. Each Supervisor also contains a Fault Manager, which determines the significance of all faults and take appropriate action to handle them. The array of unit telescopes may be divided into independently operating sub-arrays, each under the control of its own Supervisor. One obvious use-case for

this capability arises when one or more unit telescopes have been moved to new stations. The telescopes not moved form one array and each telescope that has been moved forms its own array of one telescope. Each array operates independently, the main array continuing with science observing and the other arrays doing testing and observations to compute a pointing model. The telescope operator, via the Operator module, controls the telescope primarily by interacting with a Supervisor.

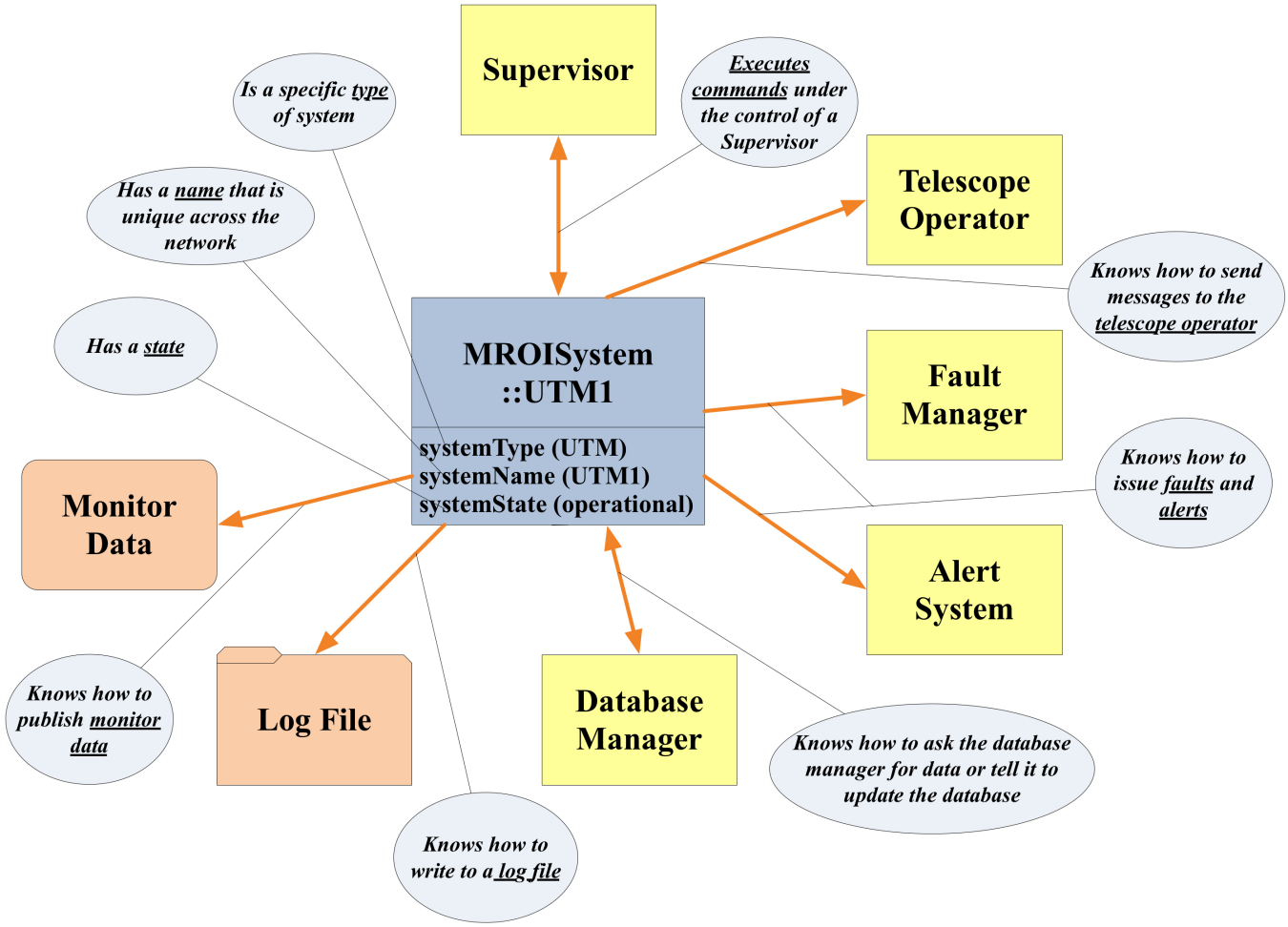

Figure 2. **Environment of an MROI Application System**

An MROI application system lives within a context provided by the Supervisory System framework. The code generation process provides this context in generating the layer of software implementing the client/server protocol. This context is described in Figure 2. An application executes commands initiated by the Supervisor, which is in charge of initializing the application and bringing it to an operational state. This application instance has a unique name and is of a specific type described in the database. The application knows how to publish monitor data and write messages to its log file. It can also interact with the database manager, as well as publishing faults and alerts. Additionally, the application can send messages directly to the computer operator. This context for an application system is referred to as the generic system interface.

The code generation framework, using the application ICD in the form of spreadsheets, generates two important software entities: (1) the client/server software layer and methods that enable the application to interact with the Supervisory System, and (2) a simulation program that enables the Supervisory System to test the contents of the spreadsheets. A set of database tools accesses the contents of the spreadsheets and stores these contents in the database, thereby providing a complete description of this type of application that is stored in

the database.

The spreadsheets describing an application contain: (1) the attributes of the system as a whole, including links to reference documentation, (2) a detailed description of all monitor data produced and what is to be stored permanently in the database, (3) a description of all fault conditions and what to do about them, and (4) commands for controlling the application together with any parameters those commands take.

An interferometer must manage both object types and instances and this distinction is reflected in the database. This is one of the key differences between an interferometer and a single dish telescope. On a single dish telescope there is one telescope mount system, for example. On the MROI interferometer there are 10 mount systems. There is one type (UTM) and 10 instances of that type. Each instance requires different sets of initialization data. Much confusion results from failing to distinguish between types and instances.

Integrating a new type of application into the MROI software system is a fairly straightforward procedure. First, the system definition spreadsheets must be filled out. These define the public interface to the system. Then the application software is added to the MROI build system. Adding Java systems to the build system is simple but C systems are sometimes more difficult to integrate into the build process. This step generates the client/server software layer and adds the system definition entries into the database. There is a subsequent step that defines application instances in a particular system configuration, but this step will be discussed in the next section.

The integrated MROI software system is shown in Figure 3.

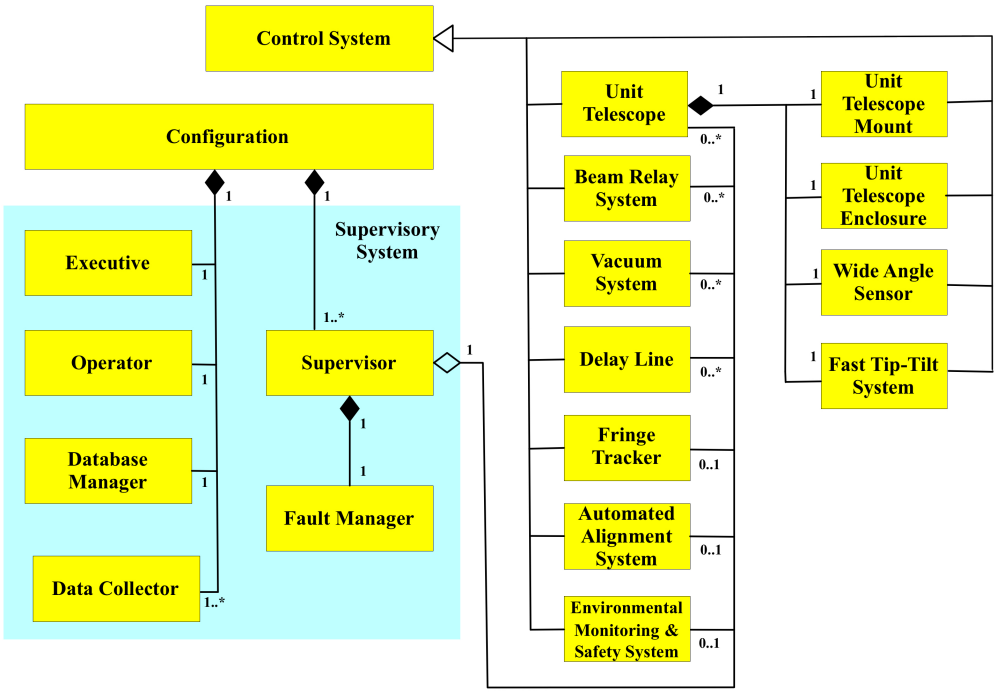

Figure 3. **MROI Software System as a Whole**

4. MROI DATABASE

The general structure of the MROI database is depicted in Figure 4. It is divided into five sections, where a section is merely a group of logically related database tables. The System Definition Section contains the contents of the application ICD spreadsheets described in the previous section. The Configuration Section describes a set of unit telescopes and application systems used in operating the system. The Published Data Section houses all the monitor and science data collected during the operation of the interferometer. The Execution History Section contains data on when applications and configurations have been executed, as well as execution logs, state history, and a record of faults and alerts. The final section is the Science Project Section, which contains data about approved science projects and links to published data and execution history. All sections of this database have been defined and are operational with the exception of the Science Project Section, which is only a skeleton at this time.

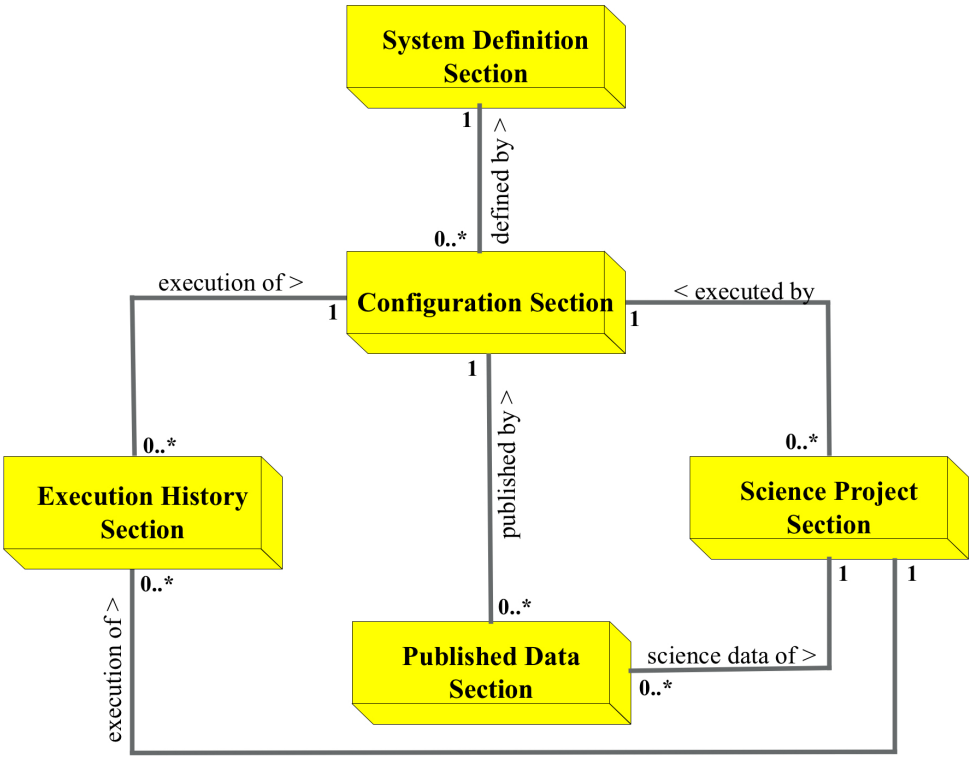

Figure 4. **Structure of the MROI database**

The Configuration Section is the most complex and describes a set of configurations. Each configuration is uniquely named and its execution is tracked over time. A configuration contains a detailed description of all application instances, including all data used to initialize the application instances, the computers they run on, and what hardware they may be associated with. For example, the pointing model for a particular unit telescope is stored in a table that references that particular unit telescope and the particular station on which it resides. The Configuration Section of the database requires 23 database tables. Altogether there are 53 tables in the database.

The MROI Supervisory System and its database definition are designed to accommodate both a testing and production environment. To accomplish this goal, we have a flexible system for specifying a configuration and storing it in the database. We use spreadsheets to populate the set of configuration tables. First, there is a spreadsheet that identifies all computers in the network, their basic characteristics, IP addresses, etc. Next, initialization data are specified and can be both text-based or binary data loaded into the database from files. Each particular set of initialization data is uniquely named within the database. Application instances are then defined, including instance names and system types, which computer they run on, ports on which they listen for

clients and publish data, the sets of data that initializes them, and any hardware or other application with which they are associated. These application instances also include components of the Supervisory System. Finally, a configuration spreadsheet specifies one or more uniquely named configurations which specify the Supervisory System components and each array together with the Supervisor/Fault Manager and the applications they manage. The particular version of the software build system is also specified as part of a configuration. For a production environment all aspects of these spreadsheets are carefully checked, however a testing environment operates under more relaxed requirements. These spreadsheets are loaded into the database to define a particular runtime environment. In a production mode, we expect the telescope operator, via a GUI, to pick a pre-defined configuration from the database, perhaps update that configuration by making minor modifications, and start the system using that configuration.

The database tables in the System Definition Section are loaded into the database from the spreadsheets that define a particular system. They must be loaded prior to any use of those subsystems. The Configuration Section tables must be loaded next and prior to any use of those particular configurations. The Published Data Section and Execution History Section database tables are loaded by the Supervisory System as a configuration is executed. The Science Project Section, when it is implemented, will be loaded from the proposal system and updated by the Supervisory System as projects are executed.

5. TESTING THE SUPERVISORY SYSTEM

The MROI Supervisory System is packaged into a standalone project in a Git repository, which we refer to as Core. This Core project includes any third party libraries in the form of jar files. Our standard testing environment is a Linux platform using Debian with a standard set of software development tools. The Core project can easily be built within such an environment. A full testing framework consists of adding applications to this Core project, creating a test database, and populating that database with application definitions and configurations, as previously discussed.

Testing the MROI Supervisory System is challenging. Three levels of system-wide simulations have been considered. The first simulation level is a test framework that is automatically generated from the spreadsheets characterizing a subsystem, allowing that framework to collect data from the subsystem and store it in the database, as well as execute all commands associated with that subsystem. This level is designed to test how the Supervisory System interacts with a particular subsystem. The second simulation level executes a detailed simulation of the actual hardware in that subsystem. For example, we have a detailed simulation of a Unit Telescope Mount. This second level allows one to test the functionality of a subsystem and its integration with the Supervisory System. These two simulation levels have been implemented and are fully functional. A third simulation level is possible: a system-wide simulation of collecting astronomical data based on executing observing projects. This third simulation level, which is much more complex, has been considered and may be implemented in the future. These levels of simulation software will be used extensively in the construction phase of the interferometer, but we also expect them to be a permanent part of the MROI operations as future enhancements are developed.

The first simulation level may sound trivial but it is not. Its main goal is to test the interface to a subsystem from the Supervisory System's perspective. It is a complete test of the ICD spreadsheets that define the relationship between the Supervisory System and that subsystem. In this case the code generation framework generates the layer of client/server code for the subsystem as previously discussed, but also generates a test program that implements all of the public methods required by that subsystem. This set of methods includes functions that generate monitor data of the proper type, including pixel arrays of varying sizes, and also commands with the appropriate parameters. It is important to understand that these programs produce data that are acquired by the Data Collectors and stored in the database in the same manner as in a real operation. The generated program is quite sophisticated, with run-time parameters that can alter the sizes of pixel arrays and their production rates, thus enabling one to test data acquisition at high rates. Likewise, commands contain run-time parameters that can specify a sleep-time that simulates a function that may take a significant amount of time to complete. In this simulation mode, the data being stored is meaningless and the commands do nothing more than log the time together with the parameters input to them. Nevertheless, this mode is a complete test of the interface to this subsystem from the Supervisory System's perspective.

This testing mode can also be used as a vehicle for testing features of the Supervisory System itself. The generated programs only depend on the spreadsheet ICD. So an ICD can be crafted to test arcane features of the Supervisory System that may not be present in all subsystems. For example, we have a spreadsheet that tests the ability to produce and store monitor data of all supported data types as well as arrays of those types. We will maintain a suite of such test programs to be used in regression testing.

The second level of simulation is familiar to most telescope environments. This level simulates the hardware functionality associated with that subsystem. Its value is directly related to the fidelity with which the software simulates the actual hardware. Currently, we have a rich simulation of the telescope mount in a unit telescope. We can also operate the fast tip-tilt system in a unit telescope in a simulation mode. We hope to add the unit telescope enclosure system to this list as well. Obviously, this simulation level is designed to test the functionality of that subsystem. Since the ability to collect and store monitor data is tested in simulation level one, the principle functionality that is added by this second level is testing the actual commands associated with that subsystem and operating that subsystem in a realistic mode.

Currently, the MROI software project is in a development and testing mode. We have one telescope on site and are preparing for formal project verification milestones using the systems within a unit telescope under the control of the Supervisory System. The Executive, Database Manager and Data Collectors are fully functional. Additional work to be done includes enhancements to the Operator interface, Supervisor, and Fault Manager. At present, the Supervisor is capable of executing scripts initiated by the Operator, but only in an elementary fashion. The addition of a rich scripting environment will be one of high priority items for the near future. The current simulation environment is capable of simulating an array of 10 unit telescopes. As additional subsystems are added, e.g. the Delay Line and Fringe Tracker systems, we expect to implement the second level of simulation for each of those subsystems.

ACKNOWLEDGMENTS

This material is based on research sponsored by Air Force Research Laboratory (AFRL) under agreement number FA9453-15-2-0086. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and or the U.S. Government.

REFERENCES

[1] D.F. Buscher, M.J. Creech-Eakman, A. Farris, C.A. Haniff, and J.S. Young. *The Conceptual Design Of The Magdalena Ridge Observatory Interferometer*. Journal of Astronomical Instrumentation. July (2013).

[2] M.J. Creech-Eakman, C.A. Haniff, D.F. Buscher, J.S. Young, I. Payne, V.D. Romero. *The Magdalena Ridge Observatory interferometer: first light and deployment of the first telescope on the array*. Proc SPIE Optical and Infrared Interferometry and Imaging VI [10701-5] (2018).

[3] A.R. Farris, D. Klinglesmith, J. Seamons, N. Torres, D.F. Buscher, J.S. Young. *Software architecture of the Magdalena Ridge Observatory Interferometer*. Proc SPIE 7740 (2010).