2.5 Software Processes

Joseph D. Long

1. Introduction

The MagAO-X instrument depends on complex software, both in-house and external, to deliver on its science goals. To manage this complexity we have developed software processes to streamline development, track compatible versions, and apply system configuration changes. MagAO-X is, at its core, a collection of C++ applications that expose instrument capabilities through INDI[1]. The operator and observer interfaces issue commands to MagAO-X over the network, while remaining agnostic to the implementation details (e.g. particular algorithms used in the loop, versions of underlying libraries).

2. Development

2.1. Version control

Changes to the MagAO-X software are tracked with `git`, a widely-used distributed version control system. `git`, like other version control systems, manages source code changes by storing them as individual "commits" detailing what was modified. A centralized copy of this repository is maintained online on GitHub at `https://github.com/magao-x/MagAOX`.

Multiple developers can contribute to this repository, with `git` automatically merging their changes when possible or alerting developers to conflicts that need to be resolved manually. The GitHub service provides access control so that only members of the MagAO-X team can modify the central copy.

2.2. Virtual machines

MagAO-X depends on the availability of certain compilers and libraries that are difficult to install on non-Linux platforms. Even among Linux platforms, not all provide the same library packages or place files in the same locations. To simplify the contribution process for developers not running a supported Linux version, we use Vagrant[2] to automate the creation of virtual machines for developers. Within the virtual machine, software dependencies, permissions structure, and other configuration will be as on an instrument computer.

[1] https://indilib.org/

[2] https://www.vagrantup.com/

The automated setup process for the VM shares as much code as possible with the continuous integration process in the next section, and the system provisioning process discussed under Operations, ensuring it does not get out of date.

2.3. Continuous integration

To ensure all dependencies are captured in provisioning scripts, we leverage the CircleCI cloud service[3] for continuous integration (CI) testing. Every time a change is made to the MagAO-X codebase copy on GitHub, the CircleCI infrastructure attempts to build and install it in a containerized environment similar to the actual instrument OS. If the software fails to compile, or the setup scripts fail to complete, a message is sent to our Slack team chat channel to alert us to correct the error. Thus, we can always be confident that we can build and install our core software on a pristine system.

While the MagAO-X codebase has been evolving rapidly, we have found such a simple "smoke test" a good way to catch problems before running on actual hardware. If the need for, e.g., automated tests to detect regressions in software functionality arises, we can simply add a step to this CI workflow.

2.4. Code quality metrics

The MagAO-X team uses a commercial service, Codacy[4], to identify issues with MagAO-X source code early by running automated checks on each new commit to the MagAO-X code repository on GitHub. Codacy provides a wrapper around static analysis tools for many programming languages and reports the issues identified both on a web-based dashboard and directly into our Slack team chat channel. This arrangement means that many errors are detected and quickly corrected before they are encountered in operation. The core code of the MagAO-X project has maintained an "A" rating on Codacy's code quality metrics.

2.5. Documentation

The core MagAO-X software is documented with Doxygen[5], the de-facto standard for documenting C++ code. To integrate this with user and operator guides written in Markdown or reST markup

[3] https://circleci.com/

[4] https://www.codacy.com/

[5] `http://doxygen.nl/`

language, we use Sphinx[6] (as used by the Python language and the Linux kernel documentation).

All documentation is tracked in `git` and hosted on GitHub alongside our code. New edits pushed to the `https://github.com/magao-x/handbook` repository trigger a CI build job which updates `https://magao-x.org/docs/handbook/`.

3. Operations

3.1. System provisioning and software versioning

The MagAO-X software currently runs on three computers that will be shipped with the instrument (the "instrument machines"), described in the Preliminary Design Review[7]. During the lifetime of the instrument, we anticipate that we will want to upgrade this hardware (or, at least, be able to replace failed components), and as such we need a complete description of the instrument hardware and software setup. We have documented necessary hardware configuration (BIOS settings, initial OS setup) in the handbook, and used a collection of scripts to install the fifty-odd packages used for everything from camera readout to AO loop control.

These are the same scripts exercised by CI jobs and developer VM provisioning, so we have confidence that they will not become out of date. They are stored in the same github.com/magao-x/MagAOX repository as the core code, allowing us to associate particular versions of the software with particular versions of its dependencies to aid reproducibility.

During on-sky operations, only software changes that have been committed to version control may be used to take data. (This is a policy, rather than a technical limitation, but our tools will produce warnings to that effect to reduce the chances of operator error.) Any data taken with the system will embed the git commit hash (a unique identifier tied to the contents of the change itself) for the system software and any first-party dependencies, allowing us to reproduce the state of the system at the moment the data was taken if necessary.

One advantage of a distributed version control system like `git` in operations is that connectivity to a central server is not required except to distribute changes. Should there be a lack of network connectivity at the telescope, we still retain the ability to record and (if necessary) revert our changes. Once network connectivity returns, the full history can be synchronized with the central copy.

[6] `https://www.sphinx-doc.org`

[7] Available at `https://magao-x.org/docs/handbook/appendices/pdr/`.

3.2. Logs and monitoring

The log architecture described in MagAO-X Preliminary Design Review document section 3.3 has been implemented largely as described. Telemetry data and logs are written to redundant storage (see Data integrity) and stored in a compact binary format for later analysis. In operation, log messages will be streamed over the network from the Instrument Control Computer and the Real Time control Computer to the Adaptive optics Operator computer.

3.3. Data integrity

The MagAO-X instrument machines are configured with data storage drives in redundant RAID 5 arrays, ensuring a complete bit-for-bit copy is available in the event of a disk failure while maintaining high performance. Operating system boot drives are likewise RAID 1 redundant, with either working to boot the system.

Retrieving data from the instrument will require copying it to a RAID 5 disk array and physically transporting the disks back to Tucson. (Leaving, of course, a backup copy in the instrument.) Data will be uploaded to CyVerse[8], with redundant copies stored at UA and TACC. (Maintenance of the storage infrastructure will be carried out by CyVerse.)

3.4. Recovery processes

The possibility of storage hardware failure has been mitigated with redundant storage devices, but there remains the possibility of the system software becoming corrupted through misconfiguration or user error, requiring recovery while away from the lab. To address this, the team that will be accompanying the instrument to the telescope has practiced installing and configuring the system software "from scratch", and will be traveling with a complete copy of the software packages and source code for the system on a separate storage volume.

[8] https://www.cyverse.org/