Instrument control system (ICS) suites are a continually evolving class of software packages that are highly dependent upon the design choices and application programming interfaces (APIs) of the observatory control system (OCS), as well as the hardware choices for motors and electronics. We present the ICS for MMT-POL, a 1-5 μm polarimeter for the MMT telescope, in the context of being a transitional step between the software packages developed for facility class instruments at UF such as Flamingos-II and CanariCam and in preparation for 30m-class instruments.

Our goals for improving ICS suites are to make them (a) portable (compile once, run anywhere), (b) highly modular and extensible (through the re-use of common libraries), (c) multi-threaded (to allow multiple tasks to be performed in parallel), (d) smart, and (e) easy to use and maintain. An ICS should also be well-defined and use mature languages (we choose java and python) and common standards (such as XML and the FITS file format). We also note that as hardware moves away from serial communications to ethernet, the use of TCP sockets makes communication faster and easier. Below, we present our design choices for the MMT-POL ICS and discuss our reasons for these choices and potential issues that must be addressed for future ICS suites ready for thirty meter class instruments.

**Goals of the MMT-POL ICS**

- **Portable**
  - Use of Java and Python allow for code to be compiled once and run on any architecture.
  - Ethernet TCP sockets allow agents and GUIs to be run from anywhere and connect to electronics and servers running in instrument rackmount.
  - XML is used for all configuration files.
  - Configuration changes are simple and do not require recompiling. Common changes, such as creating or modifying dither patterns, can be done live from the MMT-POL Java Engineering Console (MJEC) GUI.

- **Modular and Extensible**
  - Common tasks and data structures are combined into libraries (javaUFPProtocoll, javaUFLib, and javaMMTLib) used by both agents and GUIs.
  - These libraries are extensible for re-use in future instruments.

- **Multi-threaded**
  - Agents and GUIs are all multi-threaded so that multiple tasks can be performed in parallel.
  - Performing tasks in parallel increases efficiency but synchronization must be carefully implemented to avoid collisions or deadlock.

**Goals (continued)**

- **Smart**
  - Agents and GUIs work together to automate complex tasks, such as observations.
  - The executive agent has 14 active threads running during an observation to coordinate the various agents and GUIs.
  - An observation set entails dithering the telescope to X positions, cycling through all 4 half-wave plate positions Y times at each, commanding the detector to take an image Z times at each position, and collecting telemetry data from the various agents to be added to the image header of each exposure.
  - This is all performed with one click of the **Observe** button after setting a few parameters.

**Easy to use and maintain**

- Everything is run through MJEC. Its main tab (Fig. 3) displays all commonly changed parameters along with important health, status, and telemetry data.
- An easy-to-use quick look display tool (JDD) can connect to the image server from anywhere, providing a real-time view of the data. It can store multiple buffers and offers tools such as line cuts, stats, arithmetic between buffers, zooming, various scaling algorithms, and color maps.

**Software Issues for 30m class Giant Segmented Mirror Telescopes (GSMTs)**

Each instrument at a GSMT would communicate with the Observatory Control System (OCS) through a middleware layer (software communications backbone, SCB).

- Agents and GUIs could be built upon existing common libraries developed for MMT-POL combined with observatory-defined APIs to communicate with the OCS.
- Big issue (1): Implementing APIs to use the SCB to communicate with the OCS.
- Big issue (2): Data Handling Client must be able to process vast amounts of data quickly (10 Gbit/s ethernet would be optimal) and use relational database management systems (RDBMs) to mine huge databases of meta-data.

**References & Acknowledgements**

Packham et al (2010, SPIE, 7735E, 215P)