COORDINATING SIMULTANEOUS INSTRUMENTS AT THE ADVANCED TECHNOLOGY SOLAR TELESCOPE

S. Wampler, B. Goodrich, E. Johansson, National Solar Observatory, Tucson, AZ, U.S.A

Abstract

A key component of the Advanced Technology Solar Telescope control system design is the efficient support of multiple simultaneously operating instruments sharing the light path provided by the telescope optics. The set of active instruments varies with each experiment and possibly with each observation within an experiment. The flow of control for a typical experiment is traced through the control system to present the key aspects of the design that facilitate this behavior. Special attention is paid to the role of ATST's Common Services Framework in assisting the coordination of instruments with each other and with telescope motion.

INTRODUCTION

The Advanced Technology Solar Telescope (ATST) [1] is being constructed atop Haleakalā on the island of Maui, Hawaii. With a 4m off-axis primary mirror, ATST is the world's largest solar telescope and includes fully integrated adaptive optics, an unvignetted light path, a 16m rotating Coudé lab and numerous other novel features. ATST is designed to be operated as a laboratory with multiple instruments sharing the light path. Each instrument includes one or more science cameras, each delivering data to the ATST Data Handling System (DHS) at up to 1GBps [2]. During observations, the active set of instruments must cooperate and coordinate with telescope motion and light-path adjustments.

The ATST control system manages this cooperation and coordination using a pair of principal systems: the Observatory Control System (OCS) and the Instrument Control System (ICS). The ICS manages the coordination of the individual instruments while the OCS coordinates the Telescope Control System (TCS) actions with the ICS. This separation allows the OCS to focus on large-scale coordination of an entire experiment (for example, instructing the ICS to prepare instruments while the TCS is active) without having to deal with the additional complexity of managing a cooperating set of instruments. Meanwhile, the ICS focuses on the coordination of the active instruments within an individual observation without having to deal with the coordination of telescope motions with those instruments.

All of the ATST *principal* software systems (OCS, ICS, TCS, and DHS) are built using the ATST Common Services Framework (CSF) [3] which provides significant support for their operation in a highly distributed environment. CSF is built on a container/component model with containers providing essential services and lifecycle management to application components. Components are the fundamental building block for all ATST applications. A special form of component, the *controller*, implements a command/action/response model for device control [4] where configurations describing changes of state are submitted to controllers. Controllers then act upon these configurations and, sometime later, report their success or failure in achieving the target state. Services are provided based as much as possible on need and not implementation. For example, CSF provides a middleware-neutral interface for communications among components. CSF also provides substantial *technical architecture* support (how something must be implemented to integrate into ATST software), allowing software developers to concentrate more on the *functional behavior* of their code (what needs to be done to accomplish a task). The topmost layer of CSF, *Base* [5], maps the generic service framework provided by the lower levels of CSF into a technical architecture more tailored towards the actions performed in observatory operation. Base provides motion controllers, hardware connection services, and numerous other pieces commonly needed when controlling mechanical devices. Often a software developer need only adjust an existing motion controller's parameters to produce a specific mechanism control application. Typically, this simply requires adjusting some property values in a database.

EXPERIMENTS

Observing with ATST is organized into *Experiments*. An Experiment is a formal description of the behavior of ATST required when carrying out the activities associated with a scientific proposal. The Experiment includes a description of the science goal, the conditions that must be met to perform the science, what instruments are required to obtain the data, and a *Program* describing how the data is to be collected using those instruments. The OCS executes the Program to produce *Observing Blocks*. Observing Blocks are the smallest schedulable section of a Program. Each Observing Block contains a list of the instruments that are active when the block is executed, a script that coordinates the TCS and ICS actions during that execution and parameters for configuring the ICS, TCS, DHS, and the instruments for proper behavior. When the Observing Block is to be executed, the OCS runs the script using a CSF-provided script executor pool and passes all of the parameters to the script. The use of a script to manage the coordination of the TCS and ICS maximizes the flexibility available within ATST's laboratory structure. Because the script executor resides in an ATST Java container, the scripts have full access to the facilities provided by CSF, easing the development effort.

Figure 1 shows a simplified (monitoring code, checks for external cancelation, and most comments have been stripped) script used when executing an Observing Block that performs a mosaic observation around some target

Copyright © 2011 by the respective authors — cc Creative Commons Attribution 3.0 (CC BY 3.0)

```
from atst.ocs.util import Mosaics  # Other imports provided automatically

tcs = App.connect("atst.tcs")   # Establish connections to TCS and ICS
ics = App.connect("atst.ics")

# Get the list of relative positions for the specific type of mosaic
pos = Mosaics.mosaicFactory(paramSet)

# Get the current target position on the Sun.
curLoc = tcs.get(Misc.setTarget("atst.tcs", 0.0, 0.0))
target = curLoc.getDoubleArray("atst.tcs.target")
oldx = target[0]
oldy = target[1]

mosaicType = paramSet.getString("atst.ocs.script:mosaicType")
Log.note(mosaicType+' mosaic start about ('+`oldx`+','+`oldy`+'):'+
                    paramSet.getId())
icsAction2 = None
for i in range(0, len(pos)):  # Step through mosaic positions
    x = oldx+pos[i][0]        # Get absolute position
    y = oldy+pos[i][1]

    psIcs1 = paramSet.selectOnPrefix("atst.ics.")
    psIcs1.insert(Attribute("atst.ics.tcsConfigured","false"))
    icsAction1 = submit(ics, psIcs1) # Tell ICS to configure instruments

    # If not first time through the loop, must block here until previous
    #    ICS action completes before moving telescope!
    if icsAction2 is not None:
        icsAction2.waitForDone()

    # Move the telescope and block until in position.
    psTcs1 = paramSet.selectOnPrefix("atst.tcs.")
    psTcs1.merge(Misc.setTarget("atst.tcs", x, y))
    submitAndWait(tcs, psTcs1)

    # TCS is now ready, tell ICS it's ok for instruments to collect data
    #   (makeConfigROE() adds the necessary parts to the configuration.)
    psIcs2 = makeConfigROE(psIcs1)
    icsAction2 = submit(ics, psIcs2)  # More efficient to delay blocking

# Out of loop, block here until last ICS action completes
if icsAction2 is not None:
    ics2.waitForDone()

tcs.set(curLoc)             # Return to starting point
Log.note(mosaicType+' mosaic done about ('+`oldx`+','+`oldy`+')')

App.disconnect("atst.tcs")  # Close connections
App.disconnect("atst.ics")
```

Figure 1: An OCS mosaic script (Jython code).

feature on the Sun. The script interleaves and coordinates instrument actions with telescope motions for efficient observing. Note that the script does not worry about the individual instruments – that is left to the ICS. Similarly the ICS does not need to worry about telescope motions – the OCS script tells the ICS when the TCS is properly configured and when it is not.

INSTRUMENT CONTROL SYSTEM

The Instrument Control System is the principal system responsible for the simultaneous operation of the ATST facility instruments. It is a layer of software between the OCS and the instruments that implements a simple control interface for the instruments and relieves the OCS of the burden of managing them. The ICS responds to OCS commands and events, coordinating and distributing them to the various instruments as required. The ICS presents a single narrow interface to the OCS while coordinating the actions of the instrument controllers underneath it.

The ICS requires no specific knowledge about the instruments to be controlled. All information about the instruments used in an experiment and their parameter settings is passed by the OCS through the ICS, which uses a list of participating instruments to extract and forward the parameters to the appropriate instrument controllers (ICs). All ICs, which do have detailed knowledge about their underlying instruments, use the same standard narrow interface. This allows new instruments to be added without having to modify the interface or any existing ICs. If an instrument implements the ATST Facility Instrument Interface it can be controlled by the ICS. This is true for both facility instruments as well as visitor instruments.

The ICS consists of two main components: the Observation Management System (OMS) which provides the interface to the OCS, and a set of Instrument Adapters (IAs) which provide a standardized interface between the OMS and their respective Instrument Controllers. A context diagram illustrating the structure of the ICS is shown below in Figure 2.

The interface between the OCS and ICS is configuration-based and essentially stateless. Commands from the OCS are directed to the ICS using CSF configurations. The configurations are used to send the

Copyright © 2011 by the respective authors — cc Creative Commons Attribution 3.0 (CC BY 3.0)

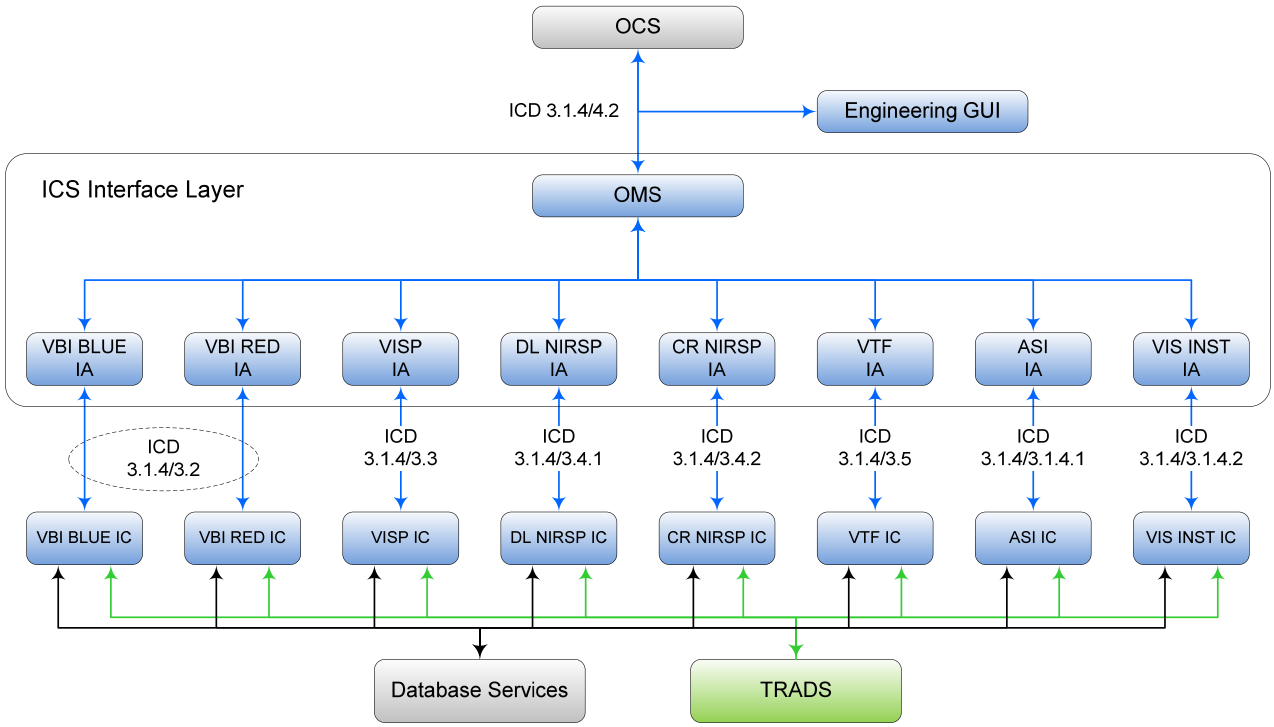

Figure 2: A block diagram of the Instrument Control System, illustrating the two main components, the Observation Management System (OMS) and the Instrument Adapters (IAs). Also shown are the Instrument Controllers (ICs) for the first generation ATST instruments, the Time Reference and Distribution System (TRADS), and the Database Services. Interfaces between the IAs and ICs define the specific observing parameters used by the instrument and passed down from the OCS.

appropriate observing parameters to the instruments, and typically include the following: a set of global experiment parameters, a list of instruments participating in the experiment, a list of those instruments that must complete their actions before the operation is considered complete, and all relevant parameters set by the user for each of the instruments. The ICS parses and interprets these configurations, extracting and forwarding the appropriate portions to each instrument in the observation. It checks on the readiness of the instruments and instructs each one to set itself up for the selected observing mode according to the parameters in the configuration. The ICS notifies the OCS when all instruments are properly setup (or not) for the observing mode.

As soon as the TCS is in position, the OCS notifies the ICS that the instruments may begin their individual observing actions for this mode (this may occur before the instruments have completed their setup activities). As soon as each instrument has completed its setup, the ICS notifies it to begin its observing actions. The ICS then monitors the instrument controllers for completion and sends status information back to the OCS.

After the OCS has notified the ICS that the instruments may begin observing, it may send one or more configurations to the ICS in preparation for the next Observing Block. The parameters in these configurations will be sent to the appropriate instruments as soon as possible, allowing them to setup for the next observing block as soon as they have completed the current one. This is referred to as *pre-configuration* and is done to maximize the efficiency of the observing process.

The ICS monitors the completion status of the participating instruments. When all of the instruments which are required to complete their observations for this observing block have completed their observing actions, the ICS terminates the observing activities of the remaining participating instruments and sends the final completion notification to the OCS.

New observing blocks are executed by repeating the pattern described above. This process repeats until all the observations for an experiment are completed or the experiment is cancelled or aborted by the operator.

INSTRUMENTS

The control of individual instruments is implemented by the ATST instrument developers. To facilitate the development of the instrument control software by these institutions, the ATST software team has developed a set of standard controllers and components that may easily be used to implement an Instrument Controller. This is called the Standard Instrument Framework (SIF). The SIF components are implemented using the Base and CSF components described earlier and consist of Management Controllers that control several other sub-controllers, Mechanism Controllers that control several Motion and Hardware Controllers, Detector Controllers that control one or more Camera systems, and an Instrument Sequencer that provides scripting support using an

Copyright © 2011 by the respective authors — cc Creative Commons Attribution 3.0 (CC BY 3.0)

embedded Jython scripting engine. Runtime access to observing scripts and parameter sets are provided through CSF database services. Motion Controllers and Hardware Controllers for control of standard motion control devices and hardware devices are provided as well. The Time Reference and Distribution System (TRADS) provides support for high resolution synchronization of instrument activity with absolute time and is the primary means of coordinating instrument mechanisms, cameras and polarization modulators with each other.

The ATST Instrument Partners are free to implement their Instrument Controllers using as desired using these tools as long as they meet the ATST Facility Instrument Interface.

Example of Instrument Control

Instruments are controlled by sending them CSF *configurations,* which are tables containing all the necessary attributes to execute a particular observing mode. As an example, the Visible Broadband Imager (VBI), which is a high-speed imager with a filter wheel, camera position and focus adjustment, might use the following attributes to execute a typical observation:

Table 1: Visible Broadband Imager Attributes

Name	**Meaning**
.experimentId	A unique ID for the experiment
.observationId	A unique ID for the observation
.obsMode	The current observing mode
.tscConfigured	A flag indicating if the telescope is in position and observing may begin
.paramSets	A list of parameter sets to be used
.paramSetSeq	The sequence in which parameter sets are to be used
.scriptName	The observing script to be run
.numCycles	The number of times to cycle through the parameter sets
.continueFlag	A flag indicating whether to continue after all cycles have been completed

A parameter set is a named collection of commonly used parameters that is downloaded from a database at run-time and passed to the script engine as an attribute. In this example a parameter set might contain the following parameters:

- The desired filter
- A map of camera field positions
- Camera setup parameters:
  - Exposure time
  - Number of frames
  - Binning
  - Region(s) of interest
- Data processing plug-ins to use

At run-time, a *configuration* with the above attributes is sent to the instrument. The instrument begins by setting all of its various devices into the proper position or mode as specified by the attributes. When everything is in position, the instrument checks to see if the telescope is in position and ready for observing to begin. If so, the observing script is run using the parameter sets and data recording commences. If not, the instrument waits patiently for the *.tcsConfigured* flag indicating it is time to start.

STATUS

The split in coordination roles, where the OCS handles high-level coordination between the TCS and ICS while the ICS handles the detailed coordination among a set of active instruments simplifies implementation of these tasks. An end-to-end simulation has been developed to test system behavior and a number of experiments have been run through this simulation, illustrating the functional correctness of this approach. The support provided by CSF at all levels of the control flow has greatly reduced the amount of specialized code that is required.

ACKNOWLEDGEMENTS

The National Science Foundation (NSF) through the National Solar Observatory (NSO) funds the ATST project. The NSO is operated under a cooperative agreement between the Association of Universities for Research in Astronomy, Inc. (AURA) and NSF.

The ATST represents a collaboration of 20 plus institutions, reflecting a broad segment of the solar physics community. The NSO is the Principal Investigator (PI) institution, and the co-PI institutions are the High-Altitude Observatory, New Jersey Institute of Technology’s Center for Solar Research, University of Hawai’i’s Institute for Astronomy, and the University of Chicago Department of Astronomy and Astrophysics.

REFERENCES

[1] See http://atst.nso.edu

[2] B. Cowan, S. Wampler, “Technologies for High Speed Data Handling in the ATST”, *Astronomical Data Analysis Software and Systems XX (ADASSXX)*, Volume 442, (2011)

[3] S. Wampler. “A middleware-neutral common services software infrastructure.” *10th ICALEPCS International Conference on Accelerator & Large Experimental Physics Control Systems*, Geneva, 10-14 October 2005.

[4] B. Goodrich, S. Wampler, “Execution of configurations using the ATST controller”, *Advanced Software and control for Astronomy*, Editors: H. Lewis, A. Bridger. Proc. SPIE 6274, 62741X (2006).

[5] J. Hubbard, B. Goodrich, S. Wampler, “The ATST Base: Command-Action-Response in Action”, in *Software and Cyberinfrastructure for Astronomy*. Editors: N.M. Radzwill, A. Bridger. Proc. SPIE7740, 77402R (2010).

Copyright © 2011 by the respective authors — cc Creative Commons Attribution 3.0 (CC BY 3.0)