Infrastructure of the Gemini Observatory Control System

K. K. Gillies and S. Walker

Gemini 8m Telescopes Project, 950 N. Cherry Ave., Tucson, AZ, 85719

Gemini Preprint # 31

Infrastructure of the Gemini Observatory Control System

K. K. Gillies and S. Walker*

Gemini 8m Telescopes Project, 950 N. Cherry Ave., Tucson, AZ, 85719

ABSTRACT

Construction of the first Gemini 8-m telescope is well underway. The software that provides the user interface and high-level control of the observatory, the Observatory Control System (OCS), is also proceeding on track. The OCS provides tools that assist the astronomer from the proposal submission phase through planning, observation execution, and data review. A capable and flexible software infrastructure is required to support this comprehensive approach.

New software technologies and industry standards have played a large part in the implementation of this infrastructure. For instance, the use of CORBA has provided many benefits in the software system including object distribution, an interface definition language, and implementation language independence. In this paper we describe the infrastructure of the Observatory Control System that supports observation planning and execution. Important software decisions and interfaces that allow Internet access and the ability to substitute alternate implementations easily are discussed as a model for other similar projects.

Keywords: software infrastructure, CORBA, planning, software standards, software design

1. INTRODUCTION

The Observatory Control System (OCS) for the Gemini 8-m Telescope is the liaison between the telescope users and the real-time software systems controlling the hardware of the telescope and instruments. In its non-real-time role at the telescope site, the OCS provides two tiers of a three tier architectural model as shown in Figure 1.

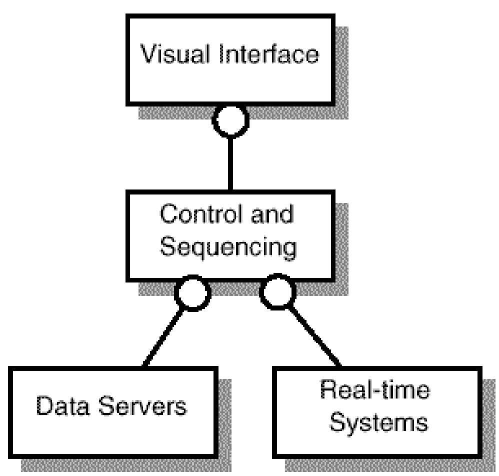

Figure 1. The OCS is a three-tiered design.

* Send correspondence to Kim Gillies.
K.G.: E-mail: kgillies@gemini.edu
S.W.: E-mail: swalker@gemini.edu

The highest level tier, the Visible Interface, presents required functionality to users of the Gemini telescopes. The Control and Sequencing tier-often called the business logic-forms the middle tier. It handles and processes requests from users, potentially resulting in accesses to the data servers in the system's third tier. When executing at one of the telescope sites, the control and sequencing tier can cause actions within the real-time systems. The architecture has several advantages that result in a software system that is flexible and easy to modify.

- The application logic in the middle tier is separated from the various interfaces that allow access to the system. User interfaces can be easily updated, or new interfaces can be added without impacting the system.
- Logic in the real-time systems and data servers is isolated in the lowest layer of the architecture, accommodating future change in those systems as well.
- The system distributes easily within the observatory and off site-even over the Internet-as needed.
- New functionality can be added without impacting the current system.

1.1. Taking a System Approach

Many OCS user requirements were written because of the importance of alleviating two problems faced by Gemini: the challenge of observing at high altitudes and the need to manage the level sophistication of the telescope and instruments. Other user requirements stress the need to support planned observing modes as well as interactive control at the telescope sites. Additionally, full and proper use of the Internet has become increasingly important over the last two years.

Initially, the Gemini telescopes will be scheduled with 50% or more of the observing time going to planned observing modes. It is hoped that the percentage of time dedicated to planned observing will increase over time. The success of planned observing depends on the quality of the scientific results, the satisfaction of the astronomers with the process, and the efficiency of the execution of the observations at the telescope.

To properly address these issues, the process of observing had to be examined so that adequate support could be provided for all the process steps. Two software goals were synthesized from this effort.

- The OCS should provide support for the astronomer from the initial proposal phase through observation execution and final data review.
- The astronomer should be able to plan his observations in detail such that another observer can execute the observation as the astronomer intended. The planning process should leave the astronomer confident that specified observations are practicable and correct.

The Gemini software system allows the astronomer to specify observations remotely or at the telescope site. The software planning tools provide a visual and interactive environment for specifying observations that is also efficient.

The science observations created by the observer are stored in a form that can be executed directly by the OCS run-time system. This new feature helps to minimize the deleterious effects of altitude, increase observing efficiency, and ensure that the astronomer's observations are executed accurately.

1.2. Infrastructure Issues

The OCS is designed and implemented as a set of cooperating, distributed objects. The objects in the system communicate using the industry standard Common Object Request Broker Architecture or CORBA[1,2]. CORBA provides many advantages in a high-level system such as the OCS.

- CORBA provides a standardized, full-featured, reliable, object-oriented, reasonably high-performance, distributed object communication infrastructure. Communicating objects can be on the same machine or across the Internet.
- CORBA provides the Interface Description Language (IDL) for specifying object interfaces. Objects implementing an interface can be written in any language that has an IDL language mapping. Standard mappings exist for C, C++, Java, COBOL and Ada2.
- A set of high level, standardized services are available that provide important functions such as object location and event distribution.
- CORBA objects communicate using a standardized protocol called IIOP. All CORBA 2.0 ORB products must implement IIOP allowing objects from different vendors to interoperate.
- CORBA objects can wrap legacy applications or entire systems allowing older code to be integrated with newer systems.
- CORBA is available commercially from multiple vendors. Several CORBA packages written in C++ and Java are freely available on the World Wide Web (See Appendix).

CORBA provides the communication infrastructure for the OCS. User interfaces use CORBA to communicate between themselves and with objects in the control and sequencing tier. CORBA objects act as wrappers for data servers providing public interfaces for local and remote user interfaces.

Most of the functionality required in the Gemini OCS is not unique to Gemini. Other astronomical telescopes have similar issues and problems. We feel the high-level design and implementation of the OCS can be applied to other telescopes systems. This paper illustrates how the OCS has been implemented by following an observation through the software system.

2. OBSERVATION DATA FLOW

In the Gemini system, the observation is the smallest scheduled entity. Observations are created during the Phase 1 process and flow through the software system. An observation has a lifecyle as shown in the following figure.

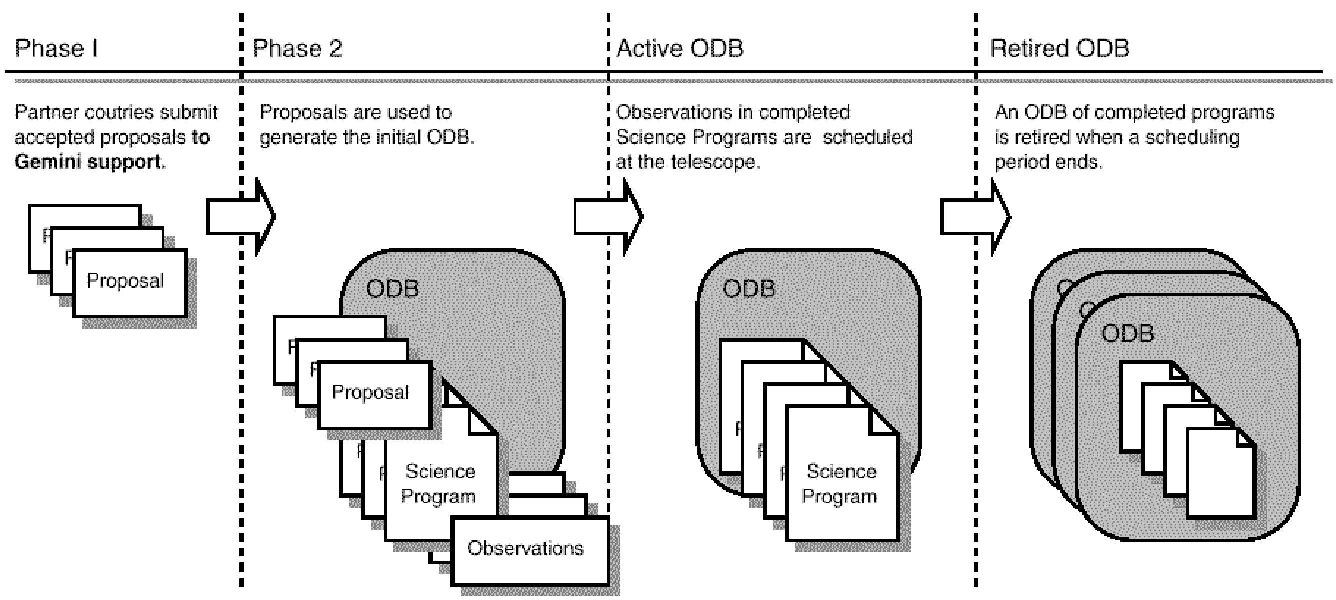

Figure 2. Lifecycle of an observation in the Gemini OCS.

2.1. Phase 1

An observation begins its life in Phase I, which is focused on the exercise of acquiring time on a Gemini telescope. Gemini does not require that specific software tools be used by the partner countries in this phase. Each partner country may have its own process for accepting and evaluating proposals for observing time, and its own set of software tools to support the process. However, the project office does specify the information Gemini proposals must contain.

Each country submits a set of successful proposals to the Gemini Project Office. An ASCII document, called the Phase 1 Program, must be created for each proposal. The format and content of this document is specified in an interface control document. This compromise allows partners to use their own tools but guarantees that successor phases will have adequate information.

Gemini is also producing a Phase 1 software tool for (optional) use by participant countries that gathers the required Phase 1 information and outputs the required document.

2.2. Phase 2

In the OCS, each Phase 1 Program is used to generate an initial Science Program. The Science Program is the document that contains all the information about the proposal, the science observations, and references to the data produced by the observations. Together, all the Science Programs form an Observing Database. There is one Observing Database per scheduling period.

At the start of Phase 2, astronomers are notified that the Science Programs are available, and are asked to complete their Science Program. Gemini provides a Science Program editor called the Observing Tool (OT) for this purpose. The Observing Tool can fetch the program from the Gemini Support Office and store the program once it is completed. During Phase 2 the astronomer can modify and update the Science Program as often as necessary.

2.3. Active Database Phase

A completed Science Program is verified for completeness and correctness by the Gemini Support Office. Once it is verified, the observations in the Science Program are available for execution. The set of Science Programs available for execution is called the Active Observing Database. Programs in the Active Observing Database can only be modified by staff members.

The Active Observing Database is used by short-term scheduling tools to produce observing plans and to suggest, at any given moment, the "best" observation for the current conditions. As observations are executed, their state is updated in the Active Observing Database, including references to the science data they generate.

The Active Observing Database resides at the telescope site, but is replicated daily at the support facility. Observers can use the Observing Tool to review the progress of their Science Program. The data references can be used to transfer the data to the observer’s site for review with local software tools.

2.4. Retired Database Phase

When the scheduling period completes, the Active Observing Database with the completed Science Programs becomes retired. The Retired Observing Database is read-only for both staff and astronomers. Retired databases may be available as part of an archive. Uncompleted Science Programs may migrate to the next Active Observing Database to allow completion or continuation.

3. THE PHASE 2 IMPLEMENTATION

The Observing Tool (OT) and the Observing Database are the two major software components involved in the Phase 2 implementation. Figure 3 shows how these two components communicate during Phase 2.

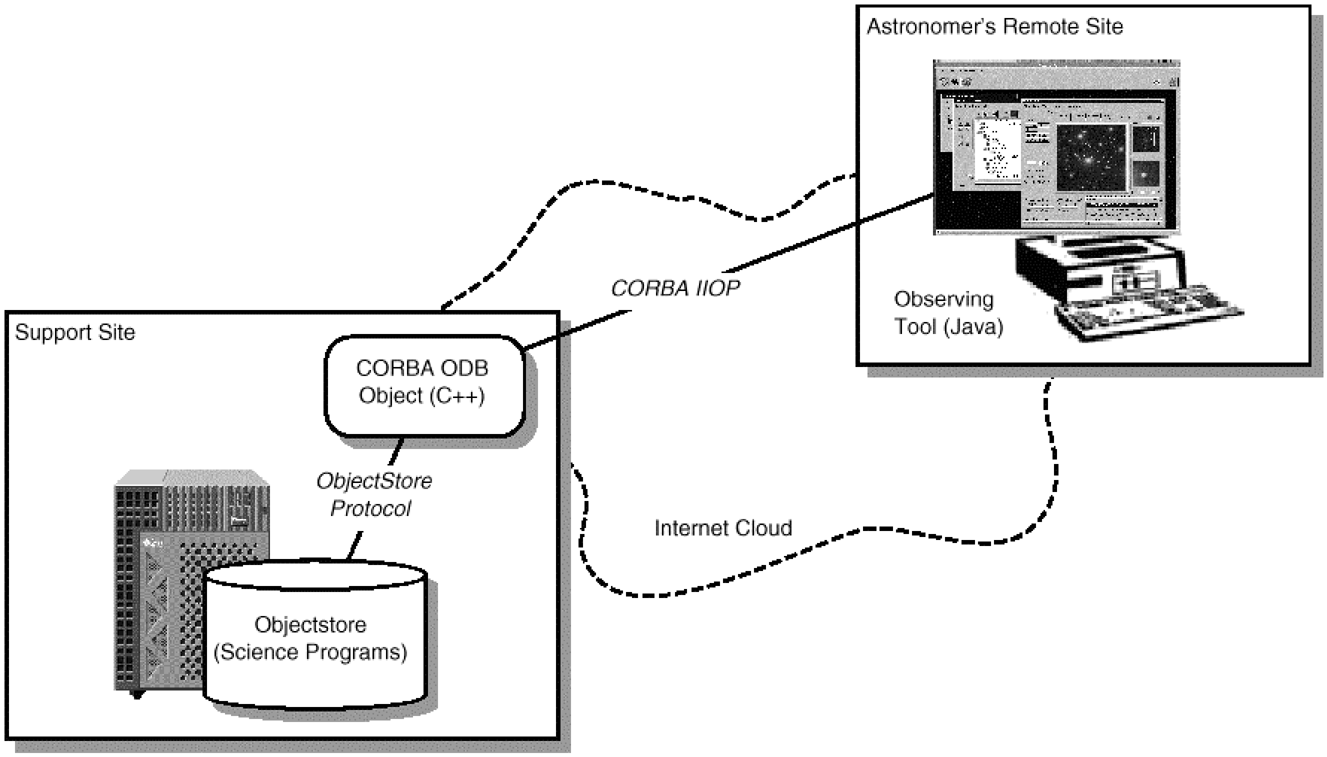

Figure 3. The Phase 2 implementation.

The OT is a Java application so the same compiled code can run on Unix systems (currently Solaris and Linux are tested), Windows NT, and Windows 95. Over a man-year of work has gone into the OT and the implementation has profited from multiple reviews and user testing. The OT provides a hierarchical, editable view of the Science Program. Sky survey images and star catalog servers can be accessed, and the output combined visually with instrument characteristics.

One part of the OT, the Position Editor, is shown in Figure 4. A full description of the Observing Tool is available in Ref. 3. Centered is a Digital Sky Survey image of the galaxy NGC1569. The image that is retrieved is the size of the range of the two peripheral wave front sensors, but a 2X zoomed image of the center of the field is shown in the figure. To the right is a pan view allowing quick movement around the entire image and a zoom view providing magnified detail. In this observation, the on-instrument wave front sensor is in use and the vignetting in the instrument is shown (the outline circle and cross). The selection of a valid guide star is facilitated by these visual aids. The outline and orientation of the detector is shown at the center (a small pointer shows the orientation). Some of these features are based on work done in European Southern Observatory's Skycat program[4].

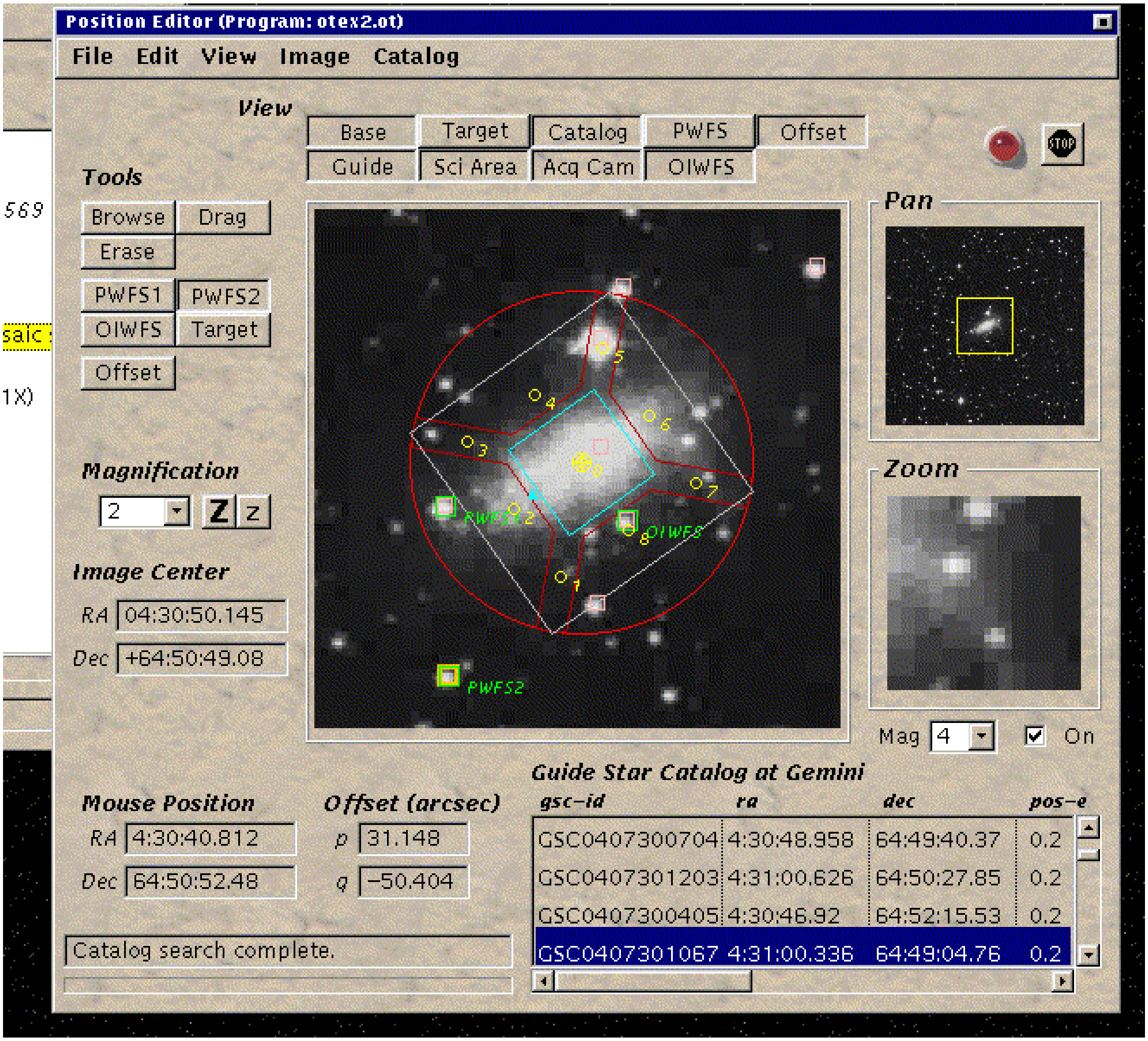

Figure 4. The Gemini Observing Tool Position Editor in action.

During Phase 2 the Observing Tool interacts with the Observing Database to store and fetch Science programs. The Observing Database is implemented as a CORBA object at the support site (and the summit). The Java-based OT uses JavaIDL, a free Java ORB implementation, from Sun.

This middle tier CORBA object is written in C++ using OmniOrb, a free C++ ORB from Ollivetti & Oracle Research Laboratory in the UK (see Appendix). It is important to note that the ORB choice is relatively unimportant since all CORBA 2-compliant ORBs use IIOP, allowing them to interoperate.

Gemini is using ObjectStore, an object-oriented database from Object Design, Inc., as the data server for Science Programs. However, the interface to the Observing Database is in no way dependent on ObjectStore; it is defined in terms of CORBA IDL, and communication between the OT and the data server is via CORBA's IIOP transport protocol. Some other database product can be substituted for

ObjectStore as long as it can support the functionality of the interface. Currently the IDL database interface consists of eight methods and around 100 lines of IDL including 40 lines of comments.

CORBA objects are located with object references that can be represented as ASCII strings known as Interoperable Object References (IOR). A standard CORBA service, the Name Service, provides a hierarchical mapping of arbitrary names to object references. The names are similar to paths in a file system. The public interface for the Name Service, also defined in IDL, provides methods for looking up references, binding references to names, and iterating over the contents of the Name Service.

When the user first makes a request to retrieve or store a Science Program, the OT contacts a known HTTP server at the support site to get the IOR of a Name Service. It then uses the Name Service to get the IOR of the Observing Database object, which, in turn, is used to fetch or store Science Programs.

4. ACTIVE DATABASE PHASE IMPLEMENTATION

The Active Database Phase is the phase of the observation's lifecyle when the observation is available for scheduling, is executed, and the associated science data acquired. During Phase 2, astronomers complete their programs and submit them to the support site where they are reviewed by Gemini staff. The verified, completed Science Programs are ready to be executed, and move to the telescope site where they form the Active Database.

Each observation in the Observing Database contains both static information such as telescope targets and instrument configurations, and dynamic information describing changes to the system configuration during the course of the observation. An example would be a 3x3 mosaic observation where at each site in the mosaic a science frame is taken in each of an instrument's filters. The telescope is offset between each of the nine mosaic locations. The observation executes at the telescope just as it was planned by the astronomer using the OT.

Several OCS user requirements influenced the Active Database Phase implementation. The OCS must allow multiple observing teams to work at the telescope simultaneously, but must guarantee that teams without the telescope beam do not affect the team that has the telescope beam. The System Support Associate (SSA) is responsible for the telescope system safety and must have the ability to monitor usage of the telescope, instruments, and resources, as well as the ability to allocate resources to observing teams.

The Observing Tool and the Observing Database are part of the implementation of this phase as well. Another software component, the Session Manager Tool, is used by the SSA to manage site resources and multiple observing teams. A component called the Sequence Executor is charged with executing a single observation. Figure 5 shows all the software components involved in observation execution; the three tier design is shown in this phase too. The roles of each software component will be more completely explained by following the sequence of operations needed to execute an observation at the site.

4.1. Sessions and Resource Management

Resources are the components of the hardware and software system that must be shared among observing teams, meaning they must be allocated by or otherwise managed by the SSA. Some examples in Gemini are the telescope beam, the three concurrently mounted instruments, the adaptive optics unit, and the comparison source unit. Most resources can be used by one team at a time.

A Session is the concept in our system that ties a set of observers, called participants, to a set of resources, and provides the observers with an environment for executing observations. Multiple sessions can run in parallel as long as the set of resources each requires is disjoint. When an observation requires a resource not held by a session, the SSA must intervene and make a decision to allocate or deny the resource request. Allocating a resource may require that the resource be taken away from another session.

4.2. The Session Object and Session Manager

An on-site observer uses the Observing Tool to request that a session be created. The OT makes contact with a middle tier object called the Session Model to make the request. The operator can then create a new Session, which puts the on-site observer in charge of determining which observations should execute in the session. The OT locates the Java-based Session Model using the Name Service and executes the methods of the Session Model interface using IIOP.

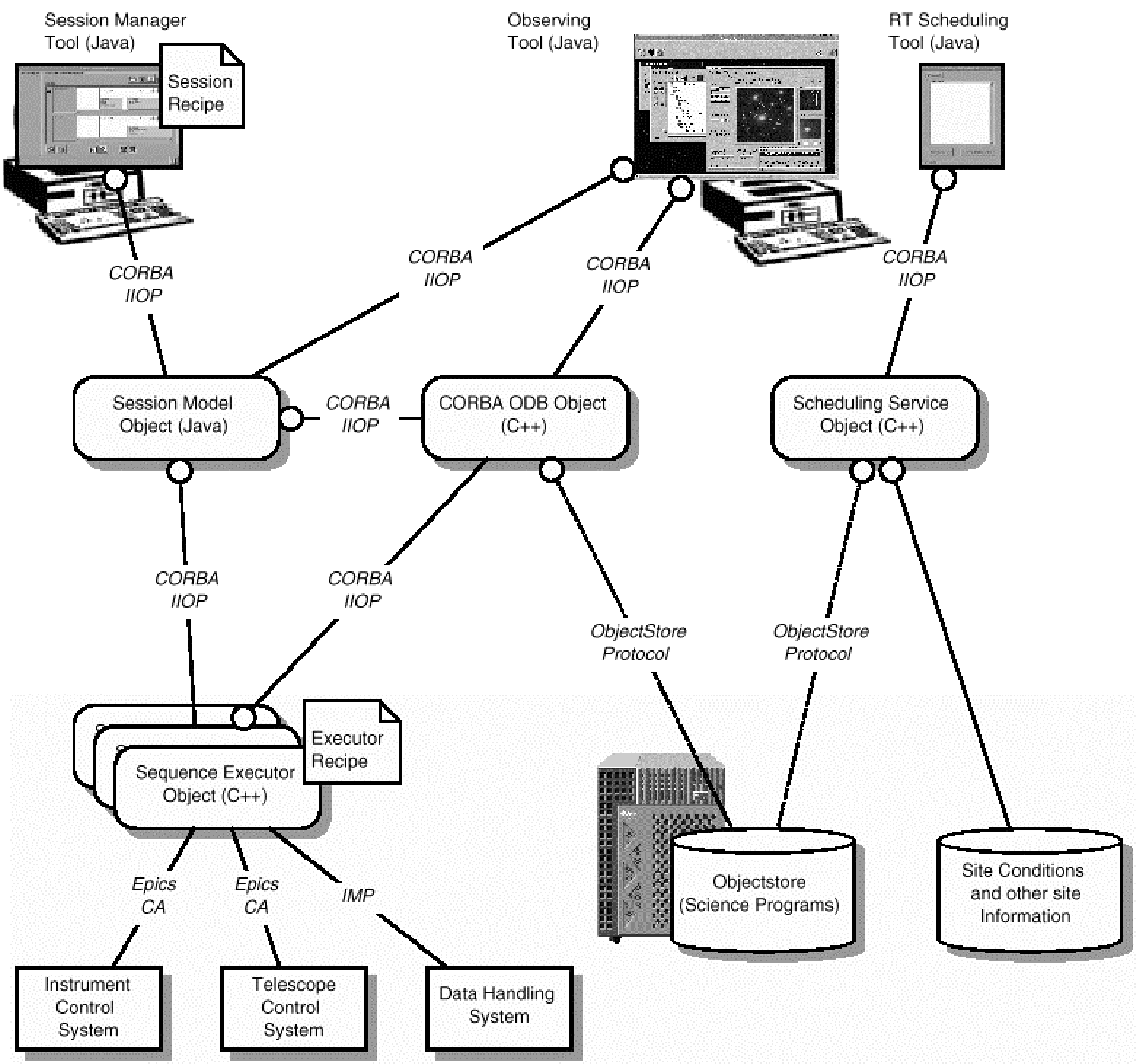

Figure 5. The Active Phase software implementation.

An off-site observer can make a request to join an ongoing session, but can't create a new session (a policy decision). By connecting to the site with the site password, an off-site observer becomes a participant who is not assigned to a session. The connected Observing Tool will be updated with low-level observatory information such as what observation is currently executing and some site status information. The SSA can assign the participant to a session, which entitles the participant to more detailed information as well as the ability to choose the next observation. We are unsure at this time how well the off-site participant features will work due to uncertainties in the quality of off-site network connections. More work will be done in this area.

The Session Model object's primary task is to keep track of the state of each of the sessions that are in progress at the site. The on-site observer uses his OT to indicate which observations should execute. The selected observations appear in the session's queue. Each session has a set of owned resources. The arrival of an observation in a session's queue triggers a comparison of its resource needs and the session's resources. If the session resources are not adequate, the observation will block until the operator intervenes.

The Session Model object is a middle tier CORBA object with a public interface that can be used by other programs. To monitor and manipulate the state of the sessions a user interface client, called the Session Manager Tool, is started by the SSA. A preliminary screen shot of this client is shown in Figure 6.

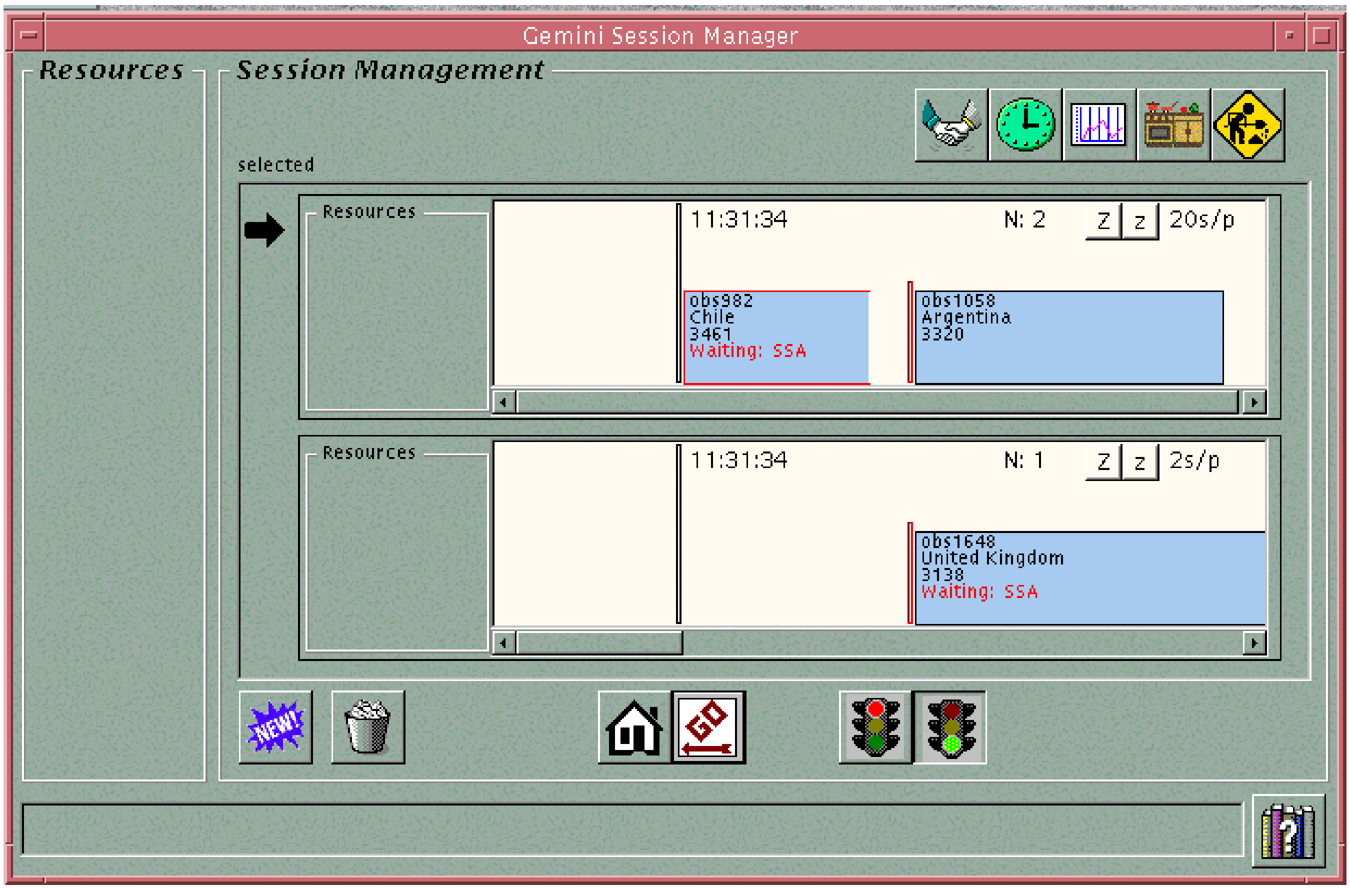

Figure 6. A preliminary screen shot of the Session Manager Tool.

The screen-shot shows two sessions in progress. The queue of observations available for execution run off the view to the right. The top session shows an observation executing. The far left area of the view will contain icons for the system resources, which can be dragged to the resource area of each session to allocate resources. This functionality is not yet implemented. Completed observations will disappear to the right as time passes, but will be available for review; a complete record of the observing session is maintained.

Generally, each observation must be validated by the SSA before it is allowed to execute. This is an observatory safety requirement. Some sessions will allow multiple observations to execute at once. Sometimes, it is desirable that resources be allocated to a session without operator involvement. Each session has a Session Recipe configuration that controls features such as these. The Session Manager Tool can vary the configuration of each session's Session Recipe in the Session Model object.

The Session Model uses and updates the observations as they pass through the session with information such as the time taken to acquire an observation's target. Gemini must keep track of how much telescope time goes to each of the participant countries. This information is folded into the scheduling algorithms. The Session Model measures this information as well as other telescope efficiency measures.

4.3. Scheduling Support

OCS provides support for the several ways that the staff may work during the planned observing blocks. The OT supports the creation of Science Plan documents that consist of observations from different Science Programs. In this model, an on-site observer will expect certain conditions at the site and plan his night ahead of time by creating a plan and storing it in the Observing Database. He might have more than one plan ready for different condition sets. At the site, the plan is opened and used to select observations for execution. New observations, primarily calibration data, can be added to the Science Plan and shared among the Science Programs of the included observations.

If conditions are changing rapidly, choosing the next observation at the site based on the measured conditions may be more appropriate. The remaining user interface in Figure 5, the Real-time Scheduling Tool, is the user interface client of the Scheduling Service Object, which provides this functionality.

The Scheduling Service Object is a CORBA C++ object with an interface that supports running the scheduling algorithms at any time. The returned result is a list of observations ordered best recommendation first. The scheduling algorithms supported by this object are described in Ref. 5. The scheduler iterates over the entire set of observations in the Active Observing Database. During each pass it applies an algorithm, builds a set of results, and a new set of surviving observations for the next iteration. At the conclusion, when all the algorithms have executed, the results are weighted and a result recommendation list is generated.

Note that in Figure 5, the Scheduling Service Object is built directly on the ObjectStore database interface to the Observing Database rather than using the Observing Database object. When this object was written we were concerned that accessing the database as vigorously as is needed by the Scheduling Service would limit the performance of the scheduling service. The scheduling object built directly on the ObjectStore interface is able to execute the scheduling algorithms of Ref. 5 on 200 to 300 observations in less than five seconds. This is much faster than is required so at some point it might be desirable to rewrite this object to use the Observing Database CORBA object.

4.4. Executing Observations

Each session in the Session Model can be associated with one or more executing observations. A Sequence Executor object is used by the Session Model to execute a single observation. The Sequence

Executor object is a CORBA object with a simple interface that allows the Session Model to start an observation, wait for completion, and interact in a few simple ways.

When the SSA commits an observation to execute, a new Sequence Executor is started and given the names of the observation and a Sequence Executor Recipe. The Sequence Executor fetches the observation from the Observing Database and begins execution of the Sequence Executor Recipe (SER).

Most observing is done as a repeatable set of steps that are followed over and over for different targets. For example: a target is acquired, instruments are configured, guide objects are validated, guide loops are checked, a final check of positioning is made, and data acquisition begins. A Sequence Executor Recipe embodies this structure of making an observation at the telescope.

Sequence Executor Recipes are written in terms of high-level Sequence Commands. These commands are independent of any particular instrument or software system. The meanings of these commands are defined in a Gemini interface control document, and the system builders are required to implement them appropriately on their systems. The sequence commands are datum, init, apply, guide, endguide, verify, endverify, observe, pause, continue, stop, and abort. (Additionally, each instrument must provide status items that reveal detailed steps of the data acquisition.) Datum and init are used during system initialization. Guide and endguide indicate that guiding is starting or ending. Verify and endverify indicate to the systems that the high-level software is entering or exiting an interactive phase. Observe tells the system to begin acquiring data. Pause, continue, abort, and stop are used to control the progress of the observation. Each system provides information that allows the OCS to determine when the sequence commands are completed.

Apply instructs a system to match a given configuration. By using configurations rather than individual commands, the OCS does not need to know the ordering of individual commands or the order of command arguments. The mapping of observations to the configurations of the TCS, DHS, and instruments is not trivial, and isn't the subject of this paper. This part of the OCS is still under development.

5. CONCLUSIONS

By using the observations planned by astronomers to drive the telescope and acquire science data, many problems are solved or minimized. The planned observing support within the Gemini OCS is a good example of how CORBA can be used to provide a flexible, extensible, maintainable software system for semi-automatic execution of observations. We believe our system design can be easily adapted by other observatories with requirements similar to Gemini's.

APPENDIX. SOURCES OF CORBA INFORMATION

There is plenty of available information on CORBA on the World Wide Web. A web page listing free CORBA information is at http://adams.patriot.net/~tvalesky/freecorba.html. The C++ ORB used by Gemini, OmniOrb2, has a web page at http://www.orl.co.uk/omniORB/omniORB.html. JavaIDL is available from http://www.javasoft.com or http://www.sun.com. The Object Management Group, the standards group responsible for CORBA, has a web page at http://www.omg.org.

REFERENCES

1. A. Pope, *The CORBA Reference Guide*, Addison-Wesley, 1998.
2. *CORBA/IIOP 2.1 Specification*, OMG Document 97-09-01, 1997. Available at http://www.omg.org.

3. S. Wampler, K. Gillies, P. Puxley, S. Walker, "Science planning for the Gemini 8m telescopes," *in Telescope Control Systems II*, H. Lewis, ed., *Proc. SPIE* **3112**, pp. 246-253.
4. M.A. Albrecht, A. Brighton, T. Herlin, P. Biereichel, and D. Durand, "Access to Data Sources and the ESO Skycat Tool," in *Astronomical and Data Analysis Software and Systems*, G. Hunt and H.E. Payne, ed.*, Astronomical Society of the Pacific Conference Series* **125**, pp. 333-336.
5. P. Puxley, "Execution of queue-scheduled observations with Gemini 8m telescopes," in *Telescope Control Systems II*, H. Lewis, ed., *Proc. SPIE* **3112**, pp. 234-245.