

# The Telescope Control System of the New Solar Telescope at Big Bear Solar Observatory

G. Yang<sup>\*a</sup>, J. R. Varsik<sup>b</sup>, S. Shumko<sup>b</sup>, C. Denker<sup>a,b</sup>, S. Choi<sup>c</sup>, A. P. Verdoni<sup>a</sup> and H. Wang<sup>a,b</sup>

<sup>a</sup>New Jersey Institute of Technology, 323 Martin Luther King Blvd., Newark, NJ 07104;

<sup>b</sup>Big Bear Solar Observatory, 40386 North Shore Lane, Big Bear City, CA 92314;

<sup>c</sup>Korea Astronomy and Space Science Institute

## ABSTRACT

The New Solar Telescope (NST) is an advanced solar telescope at Big Bear Solar Observatory (BBSO). It features a 1.6-m clear aperture with an off-axis Gregorian configuration. An open structure will be employed to improve the local seeing. The NST Telescope Control System (TCS) is a complex system, which provides powerful and robust control over the entire telescope system. At the same time, it needs to provide a simple and clear user interface to scientists and observers. We present an overview of the design and implementation of the TCS as a distributed system including its several subsystems such as the Telescope Pointing and Tracking Subsystem, Wavefront Sensing Subsystem etc. The communications between different subsystems are handled by the Internet Communication Engine (Ice) middleware.

**Keywords:** solar telescopes, control systems, distributed, communication middleware, XML

## 1. INTRODUCTION

### 1.1. Overview

The New Solar Telescope<sup>1</sup> (NST) will replace the 65-cm vacuum reflector that is currently in use at Big Bear Solar Observatory (BBSO). This new telescope will feature a 1.6 meter clear aperture with an off-axis Gregorian design. Open structure design will be employed for NST to reduce the local seeing caused by the Sun's heating of the dome interior. The NST is scheduled to see its first light in late 2006. It will be the largest telescope in the world that is dedicated to solar observations once it is finished.

To fully utilize the resolving power of this large telescope, active optics and an high-order adaptive optics system will be used. The figure of the primary mirror (M1) and its alignment with the secondary mirror (M2) will be actively controlled. High-order adaptive optics will be employed to overcome the degradation of image quality caused by seeing. The temperature of the telescope will be closely monitored and adjusted to minimize temperature gradients responsible for degrading the image quality. With the help of post-processing methods such as speckle masking, we can obtain images approaching the diffraction-limited resolution of NST, which is about 0.064'' at 500 nm and 0.21'' at 1600 nm.

Large telescopes, equipped with active and adaptive optics to overcome atmospheric seeing effects like the NST, are needed to study the fine structure on the Sun because the intrinsic scales of magneto-convection structure are less than 100 km, i.e., about 0.14''. The NST, and the instruments that will be developed for it, will be of great importance to the solar physics community and are vital to the future of BBSO. The NST is a joint effort of several institutes, including the Center for Solar-Terrestrial Research at New Jersey Institute of Technology, the Institute for Astronomy at the University of Hawaii, the Steward Observatory at the University of Arizona and the Korea Astronomy and Space Service Institute.

## 2. CONTROL SYSTEM

NST is a large and complex technological system. It includes hardware subsystems that perform different telescope operations. In the normal operation of NST the telescope control system (TCS) must be able to perform the tasks listed below:

**Monitoring and controlling the dome:** TCS has to be able to open and close the dome. It has to be able to rotate the enclosure, move the shutter of the dome and follow the telescope during its operations. It has to be able to control the dome louvers with automated, proportional positioned dampers to limit wind speed in the dome interior while maximizing ventilation.

**Pointing and tracking telescope:** TCS should be able to direct the telescope to the desired target and track it for the required observation period with sufficient accuracy. It should be able to correct the cataloged astronomical coordinates of a target for precession, atmospheric refraction and telescope flexure, and apply those values to the telescope mount.

**Controlling active optics:** TCS should be able to correct the figure of the primary mirror (M1) by commanding actuators to apply the correct forces on the back and edge of the mirror. The figure changes caused by gravity and thermal effects should be corrected by either a closed-loop system or an open-loop system using a look-up table. The alignment of the secondary mirror (M2) and the primary mirror will be adjusted by controlling a hexapod on which M2 is mounted.

**Thermal control:** This function includes monitoring, reporting and adjusting the local temperature at the primary mirror, the heat stop, the secondary mirror and a number of locations in the dome interior. Its main goal is to maintain a stable local thermal environment for the main optical elements of the telescope system by controlling the active cooling systems of the primary mirror, heat stop and secondary mirror, and by controlling the dome vent gates.

**Providing weather information:** TCS should be able to collect the environmental information, i.e., the ambient temperature, wind direction and speed, humidity, pressure, etc., which will be used by other parts of the telescope system.

**Science instruments** TCS should be aware of the existence of science instruments and monitor the status of these instruments as well.

TCS for NST must contain all these control functions and should be able to coordinate them in a safe and efficient manner to fulfill the valid observation commands from the observer. It must be able to maintain the telescope in a stable and safe condition, and produce the desired images of the science target during observations.

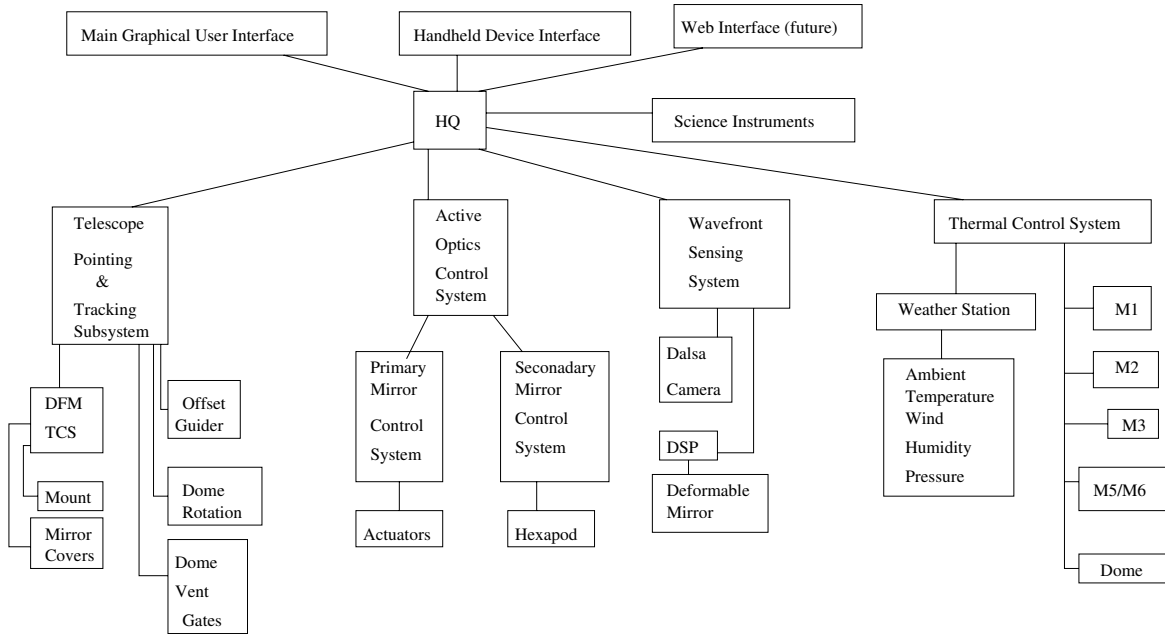
The description of the primary tasks of the telescope control system above prescribes that TCS for NST must fulfill the following requirements: (1) TCS must be able to accept commands from the operators and it should be able to tell whether the commands are valid. Only valid commands are executed. (2) TCS must control and coordinate different parts of the telescope system. These different parts might be dependent on each other and TCS must be able to resolve the control sequences; (3) TCS must be able to prevent damage caused by incorrect operations or invalid commands; (4) TCS must be able to log the status of the whole telescope system and also the commands and information produced during the observation. This information will be useful later in the analysis of the status of the telescope system, and will help the diagnosis and correction of possible control system failures.

In addition to the functions TCS has to contain, it has to be robust, reliable, flexible and easy to extend. The system must be robust and reliable because it is the core of the normal telescope operations. The system should be highly available and should be resistant to small problems and recover quickly from larger ones. The system must be flexible, extensible, and able to operate at reduced functionality even if all parts are not available. The limited size of our software development team means that not all parts will be available at first light. Furthermore, some science instruments will not be developed or ready even after the telescope control system development is finished. Therefore we need a control system that can be easily extended when new hardware or new science instruments are ready to be integrated to the whole telescope system.

TCS should be easy to operate. It should hide most of the low-level, hardware related details from the operators of the telescope. In a successful telescope control system, the operators will be able to control the telescope system by issuing high-level commands most of the time. However, in case of the system failure, we still need low level control interfaces which can be accessed to control or interact with hardware more directly and to correct problems.

### 3. DESIGN APPROACH

The TCS design will be driven by the aforementioned complexities and requirements. We will give an introduction to the design in this section. The focus will be put on the system structure and the communication considerations involved in our system.



**Figure 1.** Schematic view of the TCS structure. The whole system is divided into several subsystems which work separately, but are connected with Ethernet and are coordinated by a subsystem called headquarters. The system will also have a main graphical user interface. Other interfaces such as a web-based interface and hand-held devices are planned or are being actively developed.

### 3.1. The TCS structure

Since the telescope system is quite large and complex, it is unrealistic to put all the control functions into one software package. The effort to make such a monolithic system work and to maintain it will be insurmountable, especially considering that we are a fairly small group. Therefore we decided to make the TCS a distributed system. As parts of a distributed system, the subsystems will duplicate some features. For example, each subsystem has a GUI to allow control of the subsystem alone. Each subsystem will also need to include communication functions to contact the other parts of the system. At a first glance, this will make TCS even more complex. In reality, this actually makes the design and implementation of the system more straightforward and much easier. Figure 1 shows a schematic view of the structure of the TCS for NST.

TCS is divided into several subsystems, which include the Tracking and Pointing and Tracking Subsystem<sup>6</sup> (TPTS), the Active Optics Control Subsystem<sup>8</sup> (AOCS), the Wavefront Sensing Control Subsystem (WSCS), Thermal Control Subsystem<sup>7</sup> (THCS) and the Headquarters (HQ). TCS will provide a communication interface for the science instruments to access the telescope and to feed their status information back to the TCS. However the science instruments are not considered part of the TCS. The main graphical user interface (GUI) and the interfaces run on hand-held device or on the web are closely related to HQ and are also not considered as separate subsystems.

Among all these subsystems, TPTS, AOCS, WSCS and THCS are all low-level subsystem, which access and control hardware directly or through some controllers. These subsystems do not work independently because they may need information from other subsystems or feed information to other subsystems. Instead of making them communicate with each other directly, we employ a hierarchical control strategy. In order to make the system more flexible and extensible, we use a three-layer structure for the control system. On top of the subsystems which are responsible for controlling or communicating with the hardware directly, an HQ is added. The system-wide main GUI will also not interact with the subsystems directly. It obtains status information and sends commands to all the low level subsystems via HQ.

The low-level subsystems should be able to fulfill their jobs in an efficient and safe way as stated above. They should also report the status of the hardware that they control or interact with to HQ. The main GUI then obtains the information it displays from HQ. In case one subsystem needs information from another, it will request it from HQ. HQ will obtain

the information either by regularly polling the subsystem where it can be found or it will already have the information as a result of regular reports from the other subsystems to HQ. In normal operation, the main GUI should be sufficient for the operators to monitor the status of the entire system and issue commands to different subsystems. Each subsystem also provides an engineering interface, be it graphical or not, to provide the operators more detailed information about particular telescope subsystems.

The main GUI is the program that telescope operators use in normal conditions. An interface running on hand-held devices is also being developed. In the future, we may add a Web-based control interface. The main GUI presents the status and other information of all the subsystems. The main GUI should also provide facilities for the operators to issue simple control commands to the subsystems and to perform most normal observing tasks. Normally, the main GUI is the only place to control the whole telescope. However when TCS fails, it might be necessary to use the subsystems' control interfaces directly. HQ will act as the brain of the TCS. HQ will receive control commands or observation commands from the main GUI, analyze and divide the commands further into smaller tasks and subsequently call corresponding subsystems.

A distributed system has several advantages over a monolithic system: (1) Distributed system is more robust. Failures of subsystems may not affect the rest of the system; (2) Distributed system can also make the system more extensible. A monolithic system will be hard to maintain and every addition to the system may need a rewrite and a recompilation. A well-designed distributed system can avoid such rewrites or recompilations by providing a common interface between the subsystems and the high-level control units.

### 3.2. Choice of communication middleware

In a distributed system, the communication interface between different parts of the system is crucial to the success of the project. Considering our small software development group, we decided against writing our own lower level network communication package. Instead, we sought to use third-party, ready-to-use and high-quality technologies available for free. After investigating several technologies available, we decide to go with an object-oriented communication middleware platform. With the help of a middleware platform, instead of worrying about the details of the low-level network communication, we can concentrate on designing a powerful yet easy-to-use communication interface. A sophisticated middleware platform provides sufficient features and services that can be used in our system. There are several objected-oriented middleware platforms currently available, such as Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA) and Internet communication engine (Ice). After careful consideration, we narrowed the selections down to either CORBA or Ice as the communication platform.

CORBA<sup>3</sup> is a specification produced by the Object Management Group (OMG) that defines abstractions and services needed to develop portable distributed object-oriented applications for heterogeneous systems. CORBA defines the mechanism for providing transparent communication between applications written in different languages, running on different operating systems. CORBA allows applications to establish client/server relationships without regard to these differences. CORBA also defines a suite of Common Object Services (COS), which facilitates the development of efficient distributed systems. The first version of the CORBA specification was published in 1991. Since then, OMG has expanded and refined the specification. Despite the continuous improvement and maturing of the CORBA specifications, the implementation of the specifications did not catch up with the specifications. Although today many implementations of CORBA services in different computer languages are available, many features are still missing from some of the implementations.

Ice<sup>2</sup> is a newly-developed alternative to CORBA and other object-oriented middleware platforms. The designers of Ice learned many lessons from CORBA and made careful choice of the features and services to be provided in the Ice middleware platform. By avoiding the shortcomings and pitfalls of CORBA, and providing highly efficient and fully implementations, Ice has several advantages over CORBA: (1) Ice has a more complete feature set than any single CORBA implementation. Among the rich features and services provided by Ice, asynchronous messages and the IceGrid service are most useful to us. (2) Ice also has higher overall performance than TAO, one of the most popular and free CORBA implementations. Compared with TAO, Ice has lower latency and higher throughput\* . (3) Ice provides several language mappings, including C++, Java, Python etc., which greatly benefits software development since we can choose the most appropriate language for different situations. The source code of Ice is provided under the terms of GNU General Public License (GPL).

---

\* <http://www.zeroc.com/performance/index.html>

Because of all the benefits Ice provides, especially that Ice has all the language mappings we need, we decide to choose Ice as our communication middleware platform. Each subsystem, including HQ, acts both as an Ice client and server. A subsystem can ask for information from other subsystems through HQ. HQ can send commands to all subsystems. HQ can also poll different subsystems and/or ask them to report their status.

### **3.3. Message system**

In developing the communication interface between HQ and the different subsystems, we consider two choices. In the first choice, we can use the features and services provided by the underlying object-oriented middleware platform, i.e., Ice, to make a very elaborate and complex communication interfaces. For each command and each kind of information exchanged between HQ and subsystems, we create a method call. This might be the most efficient way to handle communications, but it makes the communication interface too complex and hard to implement. This approach means we have to put too much time and energy into developing the communication library, instead of focusing on how to implement the control functions we need. We believe this will also make the system hard to extend. If we happen to miss some commands or information required for a specific subsystem, or we add new commands, we will have to change the communication interface as well as the higher level programs to handle such changes.

Instead of making a complicated and elaborate communication interface, we can make the interface very simple, deliberately including only the necessary method calls for transferring commands, information between HQ and different subsystems. With this approach, the communication interface becomes very easy to be implemented, and we only need to specify the interface using the Specification Language for Ice (Slice), which is similar to CORBA's IDL language, and only specifies the interface, not the implementation. The developers of each subsystem can implement their own communication libraries and tailor the implementation to the needs that are most appropriate to their specific subsystem.

We decided to use the second approach. With the simplification of the communication interface, we have to use a more sophisticated message format to encode different commands and information exchanges between HQ and different subsystems. We decided to use XML, the eXtensible Markup Language, as the format of the messages. The expressive power of the XML format makes it the ideal format for such a use. The overhead of ASCII formatting of the message and the tags can be justified by the short messages and the fast network we are going to adopt. Most of the messages exchanged in TCS will be short. The longest messages containing status of the subsystems will not exceed 100 kilobytes each. The processing of XML messages becomes trivial since there are many ready-to-use libraries in different languages for this task. The processing speed is also very fast, especially considering the small message size and fast computers nowadays. It becomes very easy to extend to include more commands or information because we only need to add more messages rather than more methods. The details of the communication interface and the implementation can be found in Sergey et al.<sup>5</sup>

### **3.4. Other considerations**

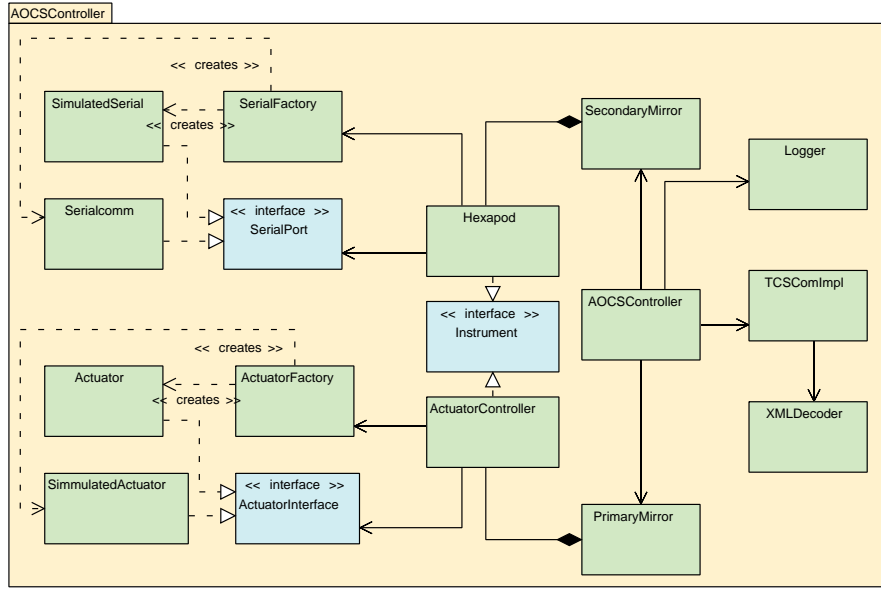
Since our developers come from different backgrounds, we do not stipulate what operating system should be used. Instead, with the help of the distributed system and Ice communication middleware, the developers of each subsystem can choose the platform that they feel most comfortable with and most appropriate for their task.

But for ease of maintenance, we decided to choose C++ and Java as the only languages for our control system. C++ is chosen because its high performance and Java is chosen because of ease of developing graphical user interfaces and its very good portability.

We encouraged the use of object-oriented analysis and design<sup>4</sup> for the subsystems. The developers are encouraged to use the Unified Modeling Language<sup>4</sup> (UML) to express their design ideas for individual subsystems. The UML consists of a set of mostly graphical descriptions for the specification and documentation of object-oriented system. UML diagrams give us a good tool for analyzing and designing the TCS software. It also enables the developers to communicate with a common language. Figures 2 gives an example of the UML class diagram for the core of AOCS.

## **4. REQUIREMENTS FOR SUBSYSTEMS**

To ensure the quality of the TCS, the requirements and functions of each subsystem has to be defined clearly. In this section, each subsystems functions will be described in detail.



**Figure 2.** UML class diagram for the core of AOCS.

#### 4.1. Main GUI and HQ

The main GUI and HQ work closely with each other. They are deemed as a single subsystem. The requirements for the main GUI and HQ include:

- The main GUI provides a display of real-time status information of all the subsystems, for example, while they are busy, waiting for completion of commands, idle, etc. It should also provide display of real-time information of the hardware that some subsystems control or have access to, such as the pointing of the telescope, weather information, etc.
- It provides control interfaces for all the functions that are necessary for normal operation of the telescope.
- Failure of the main GUI should not interrupt the operation of HQ and other subsystems and it should resume normal operation with a simple restart if HQ is still in healthy condition.
- HQ is the information and command center of the whole TCS. It communicates with the main GUI and all the subsystems. HQ collects or receives information from the subsystems. It responds to requests for information from the subsystems for normal operation of that subsystem. HQ reports to the main GUI or responds to the request of the main GUI to provide information for the user from the subsystems.
- HQ accepts commands from the main GUI, resolves the subsystems involved in the execution of the commands and sends the subsequent commands to the appropriate subsystems to accomplish tasks. If there are conflicts in the command, the main GUI is notified and the status is displayed by the main GUI. The execution results of the commands are fed back to the main GUI.
- HQ creates and manages a database that stores all the information generated by all the subsystems. The information could be used later by developers debugging the system. HQ should be able to retrieve archived information at the request of the GUI or subsystems.

#### 4.2. Telescope Pointing and tracking subsystem

The functions that TPTS handles are listed below.

- TPTS handles the pointing and tracking tasks for the NST, including the mount, dome and offset guider.
- TPTS needs to provide a “wrapper” for the control software provided by DFM Engineering, Inc., the mount manufacturer.
- Additional tasks, including flat-field operations, solar mosaics and mirror cover safety interlocks are also handled by the TPTS.
- The dome is operated by the TPTS as well. The tasks include operating the shutter, rotating the dome and operating vent gates. The vent gates may be operated by user command or according to the algorithm in the thermal control subsystem.

### **4.3. Wavefront sensing control subsystem**

The WSCS needs to fulfill following tasks:

- Gathering wave-front images from a CCD camera.
- Calculating low-order Zernike polynomial coefficients.
- Providing the wavefront error information to AOCS.

### **4.4. Thermal control subsystem**

The tasks THCS needs to fulfill include:

- Controlling the air knife and maintaining the temperature at the primary mirror.
- Providing the temperature information of the primary mirror to HQ.
- Controlling the weather station and collecting weather related information including the ambient temperature.
- Collecting information from a number of temperature sensors inside the dome and determining the desired position of the dome vent gates to improve the ventilation inside the dome.
- Monitoring the heat stop coolant.

### **4.5. Active optics control subsystem**

The tasks of AOCS include:

- Obtaining pointing information of the telescope from TPTS, wavefront error information from WSCS, and mirror temperatures from THCS at a fixed frequency.
- Calculating and adjusting the figure of the primary mirror to correct the wavefront error induced by gravity and temperature variations.
- Aligning the secondary mirror and the primary mirror by adjusting a hexapod the secondary mirror mounted on.

## 5. STATUS AND CONCLUSION

The TCS for NST is currently being developed by a small group of developers at NJIT/BBSO. Each developer is assigned one or more subsystems. A communication interface<sup>5</sup> has been defined in the Slice. All subsystems have implemented their communication interface based on this definition. A standard of communication messages has been developed. The messages that need to be exchanged between HQ and most of the subsystems have been identified.

The telescope pointing and tracking subsystem has been partly finished.<sup>6</sup> It consists of three processes, which are the command processor, kernel and engineering GUI. All the parts run on a single Linux computer. The dome control has been finished in response to the installation of the new dome in BBSO. Early versions of HQ and the main GUI have been finished using the Windows operating system. It has been tested against the TPTS system and shows good performance. An early version of the active optics control subsystem has been developed. It consists of two parts: a kernel and an engineering GUI. The control of the hexapod is working now. The wavefront sensing control subsystem is also in development. The developer has finished a GUI and the computation of the Zernike polynomials from a set of lenslet array images. The integration of the whole system will start soon. We are expecting an early version of a complete working TCS in the summer of 2006

Our experience so far has supported our decision to design a distributed system. The choice of Ice as our communication middleware platform has not only saved us much development time, but has also resulted in a more robust and extensible system. Choosing XML as the format of the messages exchanged between HQ and the different subsystems has proven to be an efficient method. The use of XML messages has no impact on the system's performance. Avoiding a propriety format has also reduced development time.

## ACKNOWLEDGMENTS

This work was supported by NSF under grants ATM 00-86999, ATM 02-36945, IIS ITR 03-24816 and AST MRI 00-79482 and by NASA under grant NAG 5-12782. We would like to thank all the groups, companies and others who provide their software libraries for free use. I would also like to thank my wife, Wei Sun, for her support.

## REFERENCES

1. C. Denker, P. R. Goode, D. Ren, M. Saadeghvaziri, A. Verdoni, H. Wang, G. Yang, V. Abramenko, W. Cao, R. Coulter, R. Fear, J. Nenow, S. Shoumko, T. Spirock, J. Varsik, J. Chae, J. Kuhn, Y. Moon, Y. Park, and A. Tritschler, "Progress on the 1.6-meter new solar telescope at big bear solar observatory", in *Advanced Software and Control for Astronomy*, H. Lewis and A. Bridger, eds., *Proc. of the SPIE* **6267**, in press, 2006.
2. M. Henning and M. Spruiell, *Distributed programming with Ice*, ZeroC, Inc., <http://www.zeroc.com/>, 2006.
3. M. Henning and S. Vinoski, *Advanced CORBA programming with C++*, Addison-Wesley, Reading, Mass., 1999.
4. C. Larman, *Applying UML and patterns: an introduction to object-oriented analysis and design and the Unified Process*, Prentice Hall PTR, Upper Saddle River, NJ, 2001.
5. S. Shumko and G. Yang, "Object-oriented communications for the NST's Telescope Control System: design and implementation", in *Advanced Software and Control for Astronomy*, H. Lewis and A. Bridger, eds., *Proc. of the SPIE* **6274**, in press, 2006.
6. J. R. Varsik, G. Yang, "Design of a telescope pointing and tracking subsystem for the Big Bear Solar Observatory New Solar Telescope", in *Advanced Software and Control for Astronomy*, H. Lewis and A. Bridger, eds., *Proc. of the SPIE* **6274**, in press, 2006.
7. A. P. Verdoni and C. Dener, "The thermal control systems of the New Solar Telescope at Big Bear Solar Observatory", in *Ground-based and Airborne Telescopes, Proceedings of the SPIE*, L. M. Stepp, eds., *Proc. of the SPIE* **6267**, in press, 2006.
8. G. Yang, "Design and implementation of the primary and secondary mirror control system for NST", in *Advanced Software and Control for Astronomy*, H. Lewis and A. Bridger, eds., *Proc. of the SPIE* **6274**, in press, 2006.