Thirty Meter Telescope: observatory software requirements, architecture, and preliminary implementation strategies

David R. Silva*[a], George Angeli[a], Corinne Boyer[b], Mark Sirota[b], Thang Trinh[c]
[a]AURA/Thirty Meter Telescope, 2632 East Washington Blvd., Pasadena, CA USA 91107
[b]Thirty Meter Telescope, 2632 East Washington Blvd., Pasadena, CA USA 91107
[c]JPL/Thirty Meter Telescope, 2632 East Washington Blvd., Pasadena, CA USA 91107

*dsilva@tmt.org; phone +1 626 395 1641

ABSTRACT

The Thirty Meter Telescope (TMT) will be a ground-based, 30-m optical-IR alt-az telescope with a highly segmented primary mirror located in a remote location. Efficient science operations require the asynchronous coordination of many different sub-systems including telescope mount, three independent active optics sub-systems, adaptive optics, laser guide stars, and user-configured science instrument. An important high-level requirement is target acquisition and observatory system configuration must be completed in less than 5 minutes (or 10 minutes if moving to a new instrument). To meet this coordination challenge and target acquisition time requirement, a distributed software architecture is envisioned consisting of software components linked by a service-based software communications backbone. A master sequencer coordinates the activities of mid-layer sequencers for the telescope, adaptive optics, and selected instrument. In turn, these mid-layer sequencers coordinate the activities of groups of sub-systems. In this paper, TMT observatory requirements are presented in more detail, followed by a description of the design reference software architecture and a discussion of preliminary implementation strategies.

Keywords: Extremely Large Telescope; observatory operations, observatory software

1. INTRODUCTION

The Thirty Meter Telescope (TMT) will be a ground-based, wide-field (20 arcmin), alt-az Richey-Chrétien telescope with a 492 x 1.44 meter segmented, 30 meter diameter primary mirror. At the start of early-science, a facility multi-conjugate adaptive optics (MCAO) system supported by a laser guide star (LGS) system will be available. This MCAO-LGS system will feed two science instruments: IRIS, a near-infrared instrument with parallel imaging and integral-field spectroscopy channels; and IRMS, an imaging, multi-slit near-infrared instrument. A seeing-limited, wide-field, multi-slit optical spectrograph (WFOS) will also be available. Concepts for six additional optical, near-IR, and mid-IR systems have been developed. A complete project summary is provided by Sanders & Nelson (2008) [1]. Descriptions of early-light adaptive optics systems and science instrumentation can be found in Ellerbroek et al. (2008) [2] and Crampton, Simard, & Silva (2008) [3], respectively.

The TMT project is currently in the midst of a 80 M$ design and development phase. Over a 25-year lifetime, the total real-year dollar value of the TMT project exceeds 2 billion US dollars. TMT will be located either in Region II of the Republic of Chile or on Mauna Kea in Hawai'i.

TMT has a collection of sub-systems that must act in concert to achieve the high-level observatory performance goals stated in the *Observatory Requirements Document (ORD)* [4] (and summarized below). Each software sub-system must be able to communicate with other software sub-systems. Minimizing total lifecycle cost (design, development, testing, integration, deployment, and maintenance) is a high-level design requirement for all sub-systems.

The Observatory Software (OSW) project is responsible for providing:

- Requirements, architecture, software, and process needed to build and integrate the entire TMT software system
- A common TMT software development framework for all software subsystems
- Communications architecture and software (middleware) for system integration

Advanced Software and Control for Astronomy II, edited by Alan Bridger, Nicole M. Radziwill
Proc. of SPIE Vol. 7019, 70190X, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.789974

- Critical software subsystems for observatory control, data management and queue observing

The OSW project will not provide all the software necessary to operate the TMT Observatory. Major sub-systems (primary control system, adaptive optics systems, instruments, etc.) will develop their own software using guidelines, procedures, environments, and tools recommended by the OSW project.

2. HIGH-LEVEL REQUIREMENTS

2.1 Starting points

High-level requirements for TMT observatory software flow from several places:

- Science-driven system performance requirements defined in the ORD [4].
- Telescope mount and optics surface control architecture defined in the *Observatory Architecture Document (OAD)* [5].
- End-to-end baseline and enhanced science operations processes defined in the *Operations Concept Document (OCD)* [6].
- Target acquisition and system configuration workflows (use cases) described in *Observation workflow for the TMT* [7].

2.2 Requirement summary

High-level observatory software requirements are described in the ORD [4] and fall into the following general categories:

- Target acquisition
- Data management and processing
- Communications (information technology) infrastructure

Briefly, detailed requirements include:

- Enable efficient user command, control, and monitoring of all observatory functions.
- Enable target acquisition and observation initiation in no more than five (5) minutes (10 minutes if an instrument change is involved).
- Enable all the baseline science operations services discussed in the OCD [4] (e.g. classical/visitor and remote observing).
- Enable the later implementation of the enhanced science operations services discussed in the OCD [4] (e.g. queue observing, observatory-based data processing pipelines, full service science archive).
- Enable telemetry capture for the purposes of performance analysis and monitoring.
- Capture and store observatory data streams (science and engineering) at mean rate of 0.02 Gbit/s (estimated peak rates: 0.10 Gbit/s). At least 100 TB of storage must be available.

In addition, all TMT software sub-systems will be required to follow the implementation process summarized later in this paper.

3. DESIGN REFERENCE ARCHITECTURE

The high-level observatory software architecture is described in the *Observatory Architecture Document (OAD)* [3].

As already discussed elsewhere (e.g. Gilles, Dunn, & Silva 2006 [8], Chiozzi et al. 2007 [9]), a standard paradigm has emerged in the last 15 years for high-level observatory software architecture. This paradigm responds to the need for an efficient observation execution core that is tightly coupled into the process for managing the end-to-end flow of information from observing proposals to data delivery, either immediately or later via a science archive. Current TMT software architecture concepts flow from this paradigm.

3.1 Observation Execution System (OES)

The functional core of the TMT software system is an observation execution system (OES). The major objectives for the OES architecture are: (i) support the control architecture defined in the OAD [5]; (ii) allow efficient observatory system configuration for scientific observations; and (iii) implement the capture, transport, and storage of all science and engineering data streams.

From a communications and integration perspective, the OES is a distributed system. It consists of a set of applications interacting with each other through a software communications (middleware) backbone. These software subsystems can be standalone software applications or they can be the software component of an integrated software/hardware subsystem. A local observatory database will provide persistent storage as necessary. A high-level view of the OES integration and communications architecture is shown in Figure 1.

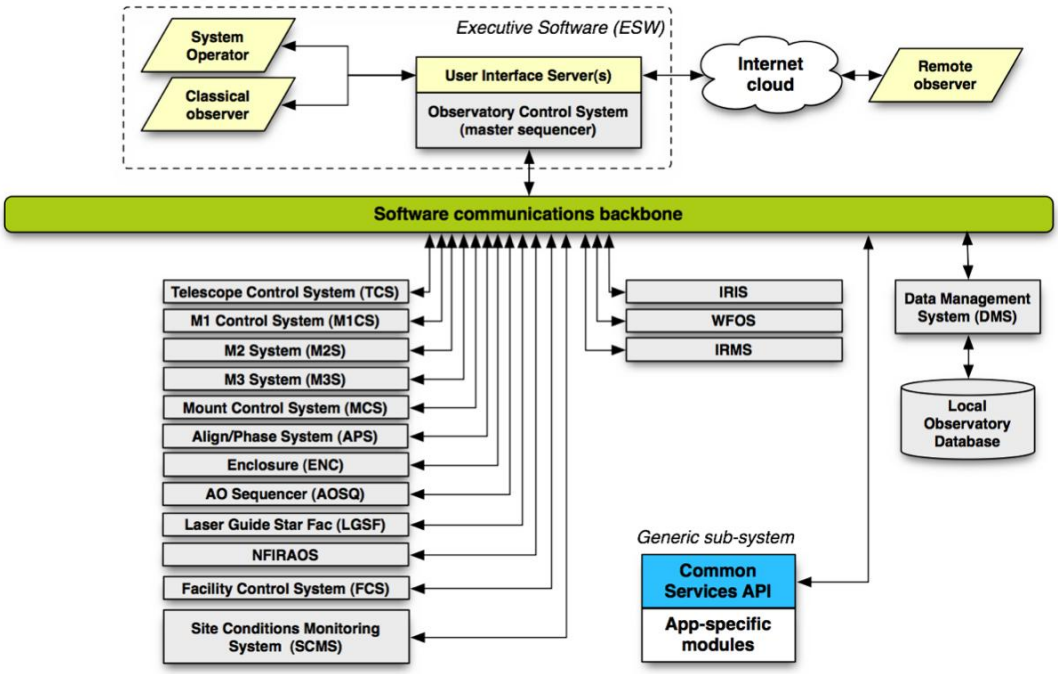

Figure 1: OES integration architecture.

The software communications backbone is an aggregate of various channels corresponding to different command, configuration, or data transfer services. The connections shown above correspond to one or more such channels. Currently anticipated services are summarized in Table 1. This list will surely evolve as detailed design proceeds. This backbone will be delivered as part of the Common Software sub-project.

It is desirable to build these services on top of available middleware toolkits such as RTI DDS, Ice, CORBA, EPICS, the Internet2 middleware initiative, FioranoMQ, etc. Various such solutions will be evaluated during the design phase. It may also be desirable that each of these services has an Application Programming Interface (API) that is service implementation neutral, i.e. it should be possible to change how a service is implemented without needing to make code modifications to the subsystems using that service. Whether such abstraction is really needed will also be evaluated during the detailed design phase.

These services will be built on top of generic protocol stack described in Table 2 and layered on top of the standards-based, physical IT infrastructure ("network"). To avoid bandwidth contention, it may be desirable to physically separate command and configuration control network traffic from science data and telemetry network traffic. However, with the advent of off-the-shelf 10 Gbit (or faster) network hardware on the horizon, such physical separation may not be necessary.

Table 1: Common services definition.

Service	Task
Location/connection	Manage inter-process communications connections
User single sign on	Manage user authentication/access control
Event	Manage event publishing/subscription
Health	Manage health check signals
Logging	Capture/store log information
Bulk data store	Capture/store bulk data (e.g. science data)
Telemetry store	Capture/store telemetry
ODB access	Middle-tier for database access
Time	Standard GPS-based Network Time Protocol (NTP) service

Table 2: Communications protocol stack.

Layer	Possible solutions
TMT-specific standards and content	
Service-specific data structure content	TBD (TMT-specific)
Service-specific data structure syntax	TBD (TMT-specific)
Industry standards	
Data structure standard	HTML, XML, FITS
Application (inter-process communication, IPC)	HTTP, SMTP, Java RMI
Transport	TCP, UDP
Network	IP
Data Link	ATM, PPP
Physical	Ethernet, CAN, ISDN, WiFi

3.2 Generic sub-system architecture

Naturally, TMT sub-systems will not be monolithic. They will be aggregates of components, including detector systems, motion control mechanisms, optical surface control actuators, integrated real-time computers, etc. Each aggregate must have some level of internal synchronization while maintaining a communications interface to the TMT software system at large. In some instances, software within sub-systems will be built using a TMT specific software development framework. In other instances, sub-system software will be implemented by an external vendor without using a TMT specified software development framework. In those cases, thin-layer adapters will be needed to integrate those sub-systems into the rest of TMT.

Figure 2 illustrates this situation. The blue connector symbolizes the global software communications backbone that ties together all sub-systems (see discussion above). The Executive Software sub-system (described further below) is the master observatory controller. Sub-system X contains software modules built using the TMT framework to control various hardware and detector components. Sub-system Y contains an adaptor module built using the TMT framework to control a vendor-supplied software component that in turn interfaces to vendor-supplied hardware components. Sub-system Z is similar to Sub-system Y but does not require a sub-system sequencer.

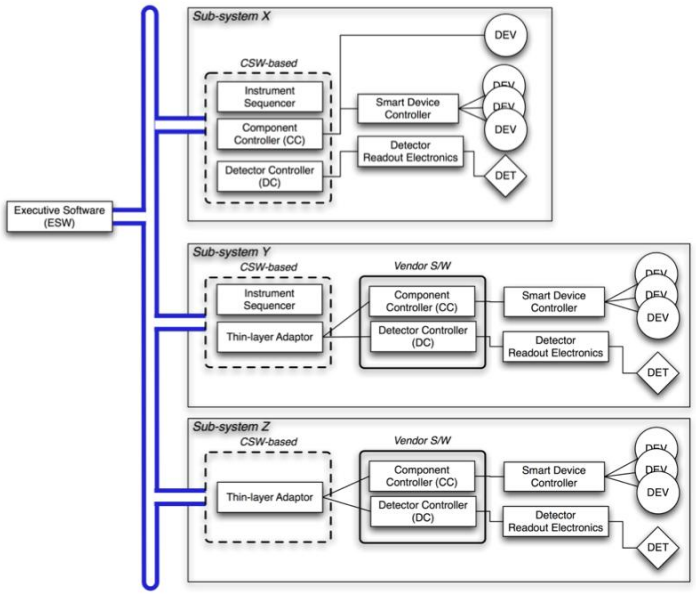

Figure 2: Examples of sub-system integration architecture concepts.

Figure 3 provides a more concrete example by presenting the existing conceptual software architecture for the early-light TMT adaptive optics sub-system. Here, the OCS ("master sequencer") module of the Executive Software (see below) manages the top-level aspects of the science observing program while the AO Sequencer manages the actions of the AO subsystems and acts as the main public interface between the AO system and the remainder of the observatory. Generic sequences include configuration (i.e. for the Laser Guide Star Facility, Narrow Field Infrared AO System, and Real-Time Controller), calibration, and operation (including managing "closing the loop" in natural or laser guide star modes). In short, control intelligence and authority is pushed down as close as possible to the actual hardware being controlled.

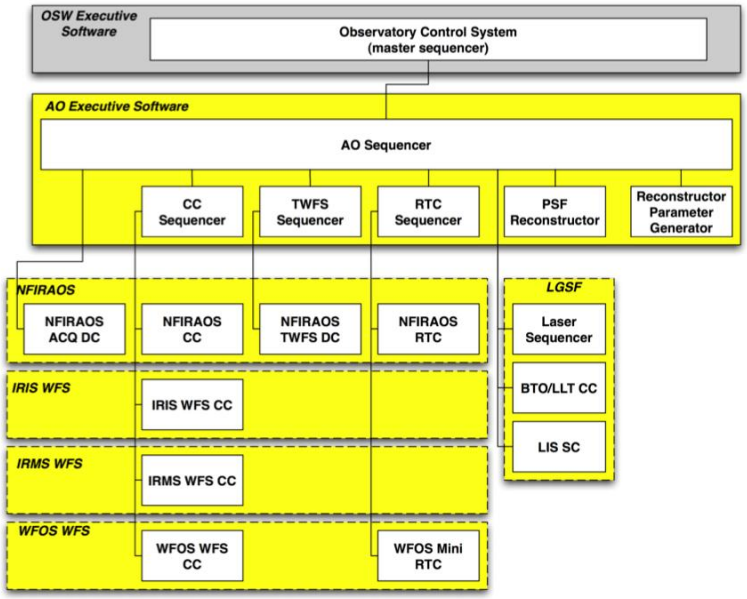

Figure 3: Adaptive optics sub-system architecture. Note that CC = component controller, DC = detector controller, WFS = wavefront sensor, TWFS = Truth WFS, and RTC = real-time controller

3.3 Hierarchical command and control

For most night-time operations, the OES command-and-control architecture is hierarchical. The transition from one system configuration ("observational setup") to another results from a sequence of activities initiated and coordinated by the Observatory Control System (master sequencer) component of the Executive Software. This coordination is accomplished in concert with a set of lower tier sequencers (as discussed above). In a limited sense, the OES command-and-control architecture is dynamic – different hierarchical relationships are established logically for different observational setups. For example, Figure 4 shows the hierarchical relationship established to execute an IRIS observation using laser guide stars.

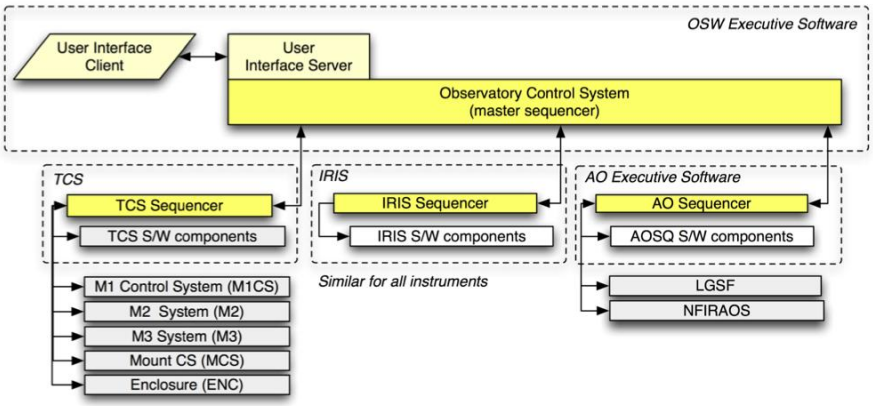

Figure 4: Control hierarchy for Infrared Imaging Spectrometer (IRIS) observation using the facility multi-conjugate adaptive optics system (NFIRAOS) with the laser guide star facility (LGSF).

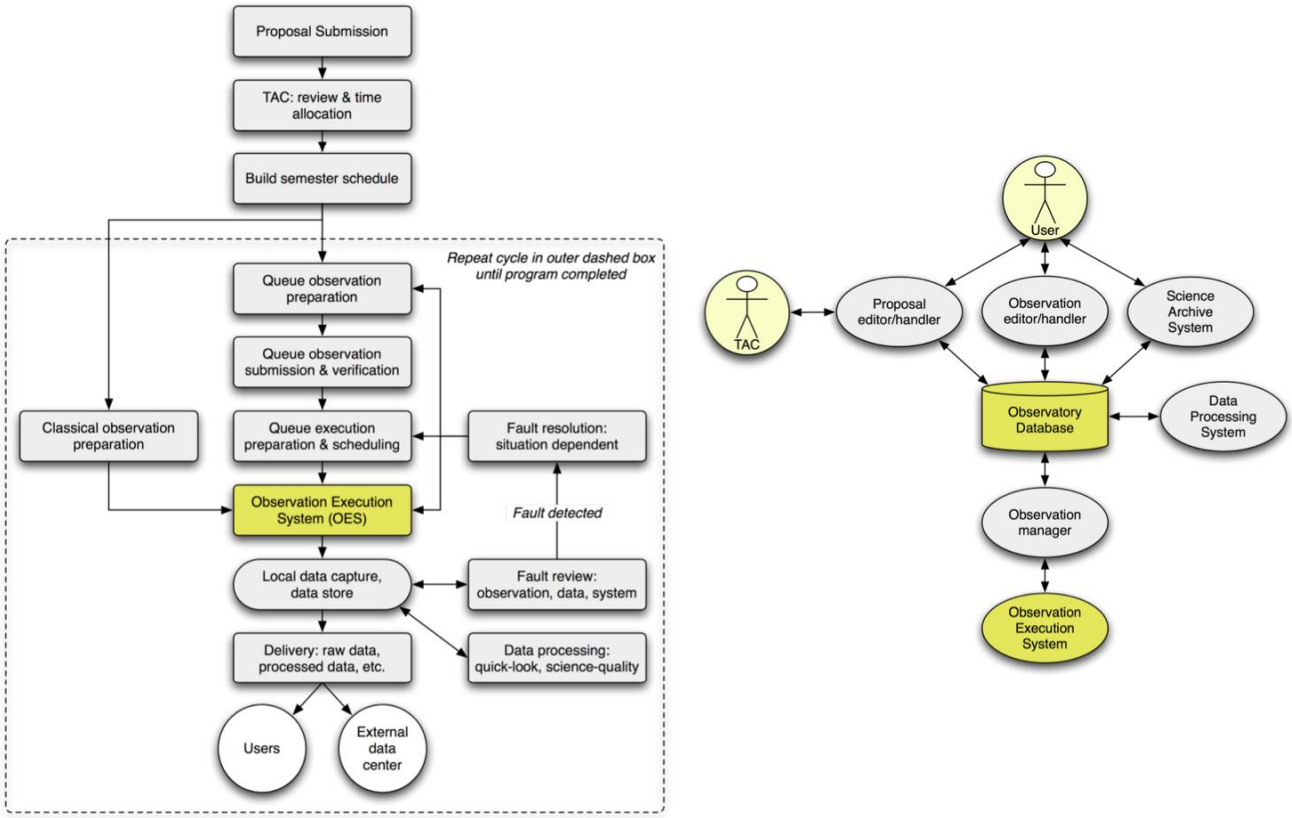

Figure 5: PES workflow (left) and conceptual architecture (right).

3.4 User interfaces

User interfaces with command and monitor functionality must be developed for classical observers (on-site or remote), systems operators, and technical staff responsible for monitoring performance. These interfaces will be graphical and have a common look-and-feel. Each user interface is essentially a thin-client interacting with the rest of the system as if it were a server. Care will be taken to allow for using such interfaces in monitor-only mode from remote locations.

3.5 Program Execution System (PES)

Observation execution is central to a larger program execution workflow common to all general-purpose observatories (see Figure 5). To support that workflow, a program execution system (PES) architecture concept has been developed (see Figure 5). Philosophically, the PES architecture is based on standard client-server principles where the observatory database (wrapped in the Data Management System, see below) acts as the central server and the PES workflow is implemented by various clients.

Due to resource limitations, the PES will be implemented incrementally. Initially, only PES workflows, interfaces, and data structures will be designed. Actual application implementation will either occur late in the TMT construction phase or during the early operations phase.

4. KEY SUB-SYSTEMS

The OSW project is responsible for delivering four key software sub-systems.

4.1 Executive Software System (ESW)

The Executive Software (ESW) enables control, monitoring, and synchronized operation of all TMT sub-systems.

Central to ESW is the master observatory sequencer that accepts observation descriptions and translates them into high-level observation sequences. These high-level sequences coordinate the actions of other TMT sub-systems involved in executing an observation. Such coordination is necessary to maximize observing efficiency via (e.g.) parallelization of tasks and hence implement the target acquisition and system configuration workflows described in Boyer & Simard (2008) [7].

ESW also provides high-level graphical user interfaces for system operators and observers, system status monitors, and environmental status monitors (external and internal). A complementary target acquisition toolkit will be provided, with support for target field visualization, basic frame algebra, and guide star selection/definition. We expect to adopt and adapt an existing solution (see Figure 6 for an example).

ESW must be designed and implemented to enable remote observing. At present, remote observing is expected to be implemented using desktop sharing based on virtual network computing (VNC).

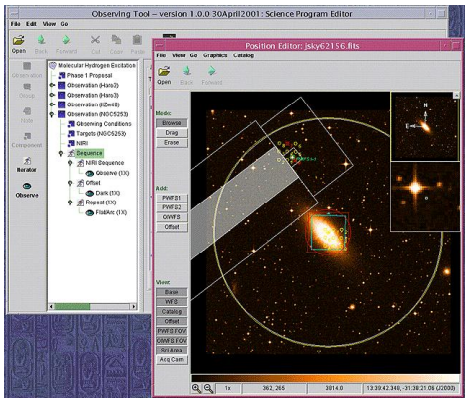

Figure 6: Gemini Observing Tool, derived from ESO JSky Java toolkit

4.2 Data Management System (DMS)

All science and (relevant) engineering data produced by TMT must be captured, time-stamped, described, stored, and (in most cases) archived. The Data Management System (DMS) combines hardware and software to accomplish those tasks.

The Data Management System (DMS) shall provide a persistent data store (with appropriate indexing and time stamping) for the following use cases:

- Science data produced by science instruments
- Telemetry data produced by technical systems
- Logging (history) data from all sub-systems
- Astronomical catalogs
- TMT software systems that require long-term data caches (e.g. APS)
- User login and contact information

These data can be separated broadly into three categories.

Update-oriented data are small (few byte to several KByte) data objects oriented towards an update-many, read-many, fast access model (e.g. status flags, mutable business objects such as observation descriptions). Data volume is expected to be tens of GB. These data will be stored within relational databases.

Read-oriented data are also small (few byte to several KByte) data objects but oriented towards a write-once, read-many, fast access model (e.g. sub-system telemetry data). Data volume is expected to be a tens of TB. For example, engineering telemetry data falls within this category.

Finally, the DMS must capture **bulk data** – large (several to many MByte) data objects oriented towards a write-once, read-many, slow access model (e.g. science detector pixel data). Data volume is expected to be at least 100 TB. These data will be written to a disk-oriented storage system with location pointers stored in relational databases. These pointers will be linked to the meta-data that describe the bulk data objects. This linkage enables the ability to query meta-data tables to locate appropriate bulk data objects and then use the pointers to find the bulk data objects in the bulk storage device.

The current DMS sub-system architecture concept is shown below.

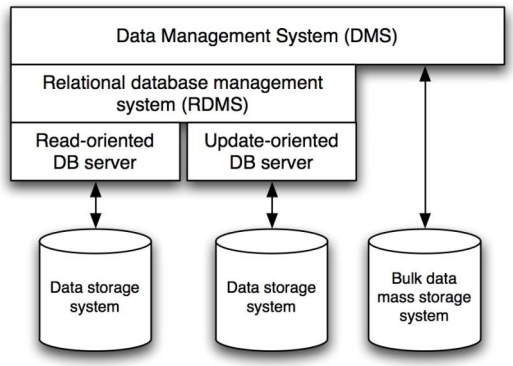

Figure 7: DMS architecture concept

A related DMS deliverable is an observatory-wide *Data Interface Control Document*. The Data ICD shall contain definitions, formats, and data rates for all TMT data and meta-data, both science and engineering data. It shall also provide high-level descriptions of the observatory database data structures. In combination, these concepts are sometimes called the observatory data model.

The DMS will also include an astronomical catalog management service.

4.3 Science Operations Support Systems (SOSS)

Science Operations Support Systems (SOSS) are a tightly coupled set of applications to manage high-level science operations workflow from proposal preparation to observation execution. Post-execution actions are covered in the Data Processing System work element (see below).

SOSS deliverables include tools to support: proposal preparation, handling, review, and time allocation ("Phase 1"); observation preparation, handling, review, and queuing ("Phase 2"); observation scheduling; and observation execution and problem resolution ("Phase 3").

SOSS implements the front-end (pre-execution) part of the PES workflow and architecture.

4.4 Data Processing System (DPS)

The Data Processing System (DPS) enables the removal of atmospheric and instrument signatures from data produced by TMT science instruments. It also enables a quick-look and long-term trending data quality assurance process.

The DPS has four main components:

- Recipes – applications that apply data processing algorithms to input data.
- Common Recipe Support Library (CRSL) – a library of procedures that establish a framework for recipe development including common command line and parameter syntax, data I/O procedures, and often-used pixel manipulation routines. The CRSL goal is to allow recipe development teams to focus on recipe intelligence (algorithm) as opposed to recipe structure. As time goes on, this library can be extended.
- Pipeline infrastructure – software system to associate science data with appropriate calibration data and recipe(s) and then execute recipe, all automatically. Intended for construction of quick-look, long-term trending, and science data product pipelines. As much as possible, pipeline infrastructure should be hardware neutral.
- Observatory quality assurance pipeline – using components above, implement quick-look and long-term trending quality assurance systems for TMT science operations.

The development of data processing workflows and requirements must be coupled to the various instrument data processing and calibration plans.

The DPS implements the back-end (post-execution) part of the PES workflow and architecture.

5. PRELIMINARY IMPLEMENTATION STRATEGIES

Concepts discussed in this section will be described more fully in the *Software Management Plan*. The SMP is currently under development. Release 1 is planned for early 2009.

5.1 Software development framework

All TMT software components will be implemented within a standard development framework, with three high-level goals:

- Adopt and/or adapt commercial and/or open source solutions that are already widely used and supported within the IT industry.
- Minimize time and effort needed to install, integrate, and verify the TMT software system on-site and make it operational.
- Minimize time and effort needed to maintain and extend the TMT software system during operations

Heavy solutions that have high, long-term maintenance or licensing costs (e.g. commercial enterprise-class middleware and libraries) will be avoided.

The standard TMT software development framework will specify development environments (OS, hardware, languages, compliers, build tools, etc) with future deployment environments in mind but not firmly set yet. For non-real-time systems, recent discussions have focused on various Unix/Linux variants in combination with Java, C, and Python. Real-time solution candidates have not been discussed extensively but the usual suspects are being rounded up: VxWorks, Real Time Linux, Java Real-Time, and LabView.

5.2 Off-site testing and integration process

A high priority goal is to deliver completely tested and debugged software to the on-site observatory. Meeting that goals requires a multi-layer testing/integration process and a central software engineering team to support that process.

Each development team is expected to develop its own set of unit tests. The exact nature of these unit tests will not be specified centrally. Successful unit testing is assumed to lead to local integration testing.

On a regular basis, development teams will submit their software to a central repository. On a nightly basis, an automatic build and integration testing process will be executed. The central TMT software engineering team will work with the relevant development teams to resolve any errors found during this nightly test.

At approximately six-month intervals, formal end-to-end system tests will be executed to verify that planned functionality has been delivered and planned performance goals met.

5.3 On-site integration

In the TMT project vernacular, Assembly/Integration/Verification (AIV) is the process that occurs on-site at the summit. All software modules must pass their integration tests at the central facility before AIV begins.

Software stored on the off-site central configuration server (defined above) shall be considered the master copy.

Software installed in the operational, on-site environment shall not be considered the master copy. Changes to software made to software installed in the operational, on-site environment shall not be considered official or permanent. All such changes shall be tagged and checked into the central configuration control server as soon as possible.

The daily build, integrate, and test cycle shall continue in the central integration facility until AIV is completed.

6. OSW IMPLEMENTATION ROADMAP

6.1 Design & Development Phase: 2008 – 2010

As discussed above, TMT observatory software grows out of a large body of existing concepts. Thus, a vast DDP effort is not needed. At this time, we expect the bulk of the high-level design and development to be completed by 2 – 3 people working for two years. Detailed design and implementation is deferred after 2010 (at least), as discussed further below.

Rather than simply impose standards and concepts, we intend to engage TMT software stakeholders closely and leverage their experience to produce a better project. All teams developing software for TMT are OSW stakeholders, including the facility AO team, the telescope optics alignment & phasing system team, the telescope controls team, and the instrument teams. To promote trust and transparency, stakeholder meetings will be held circa two (2) months before each major review to present current work and seek feedback. Whether these meetings are physical or virtual is TBD. These are working meetings with software engineers or direct software customers. Stakeholders will be invited to major reviews.

Over the next 18 – 24 months, three major reviews involving non-advocate external reviewers are anticipated: Conceptual Design, Preliminary Design, and Final Design. Roughly two months before each review, a stakeholder workshop will be held. These workshops and reviews will be based on the foundation documents shown in Table 3 (DRD = Design Requirements Document, ICD = Interface Control Document).

6.2 Implementation: 2010 and beyond

In order to develop a cost estimate, a top-down implementation plan for observatory software has been developed. This plan assumes a two-person management team, a system engineering team to support external development teams and provide system integration and testing support, a development and support team for common software, and analogous teams for the Executive Software and Data Management System sub-systems. The current estimated effort is 110 FTE, distributed non-uniformly over a seven year period. There is an initial detailed design phase with a small team that ramps up to an implementation phase and then ramps down to a maintenance and operations hand-off phase.

Again, this is not the total TMT software effort – an additional 140 FTE are currently budgeted for software development under direct TMT control, but outside of the OSW project. Additional software engineering effort is budgeted indirectly in sub-systems assumed to be delivered by vendors.

Table 3: TMT OSW design and development deliverables.

Document	Remarks
High-level Requirements	Contained in ORD [2]
High-level Architecture	Contained in OAD [2]
Software Management Plan	Development and standards framework
Software Project Plan	Budget, schedule, etc.
Data ICD	Modeled after equivalent ESO document[1]
Common Services DRD/ICD	Software communications backbone and sub-system integration toolkit
Executive Software DRD/ICD	Master sequencer, etc (see above)
Data Management System DRD/ICD	Observatory database, etc. (see above)
Science Ops Support Software DRD/ICD	PES front-end
Data Processing System DRD/ICD	PES back-end

Acknowledgements

The authors gratefully acknowledge the support of the TMT partner institutions. They are the Association of Canadian Universities for Research in Astronomy (ACURA), the California Institute of Technology and the University of California. This work was supported as well by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the National Research Council of Canada, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Universities for Research in Astronomy (AURA) and the U.S. National Science Foundation. We also thank Jennifer Dunn (NRC/HIA) and Kim Gillies (STScI) for their valuable contributions.

REFERENCES

[1] Sanders, G. H., & Nelson, J.E. 2008, The status of the Thirty Meter Telescope project, SPIE Conf. 7012, Ground-based and Airborne Telescopes, Paper 7012-45, in press

[2] Ellerbroek, B. et al. 2008, Progress towards developing the TMT adaptive optics systems and their components, SPIE Conf 7015, Adaptive Optics Systems, in press

[3] Crampton, D., Simard, L., & Silva, D.R. 2008, Early light TMT instrumentation, SPIE Proc. 7014, Ground-based and Airborne Instrumentation for Astronomy II, eds. Mclean, I. & Casali, M., in press

[4] Observatory Requirements Document (ORD), located in the Foundation Documents collection on the TMT Web home page, www.tmt.org.

[5] Observatory Architecture Document (OAD), located in the Foundation Documents collection on the TMT Web home page, www.tmt.org.

[6] Operations Concept Document (OCD), located in the Foundation Documents collection on the TMT Web home page, www.tmt.org.

[7] Boyer, C. & Simard, L. 2008,Observatory workflows for TMT, internal TMT document, TMT.AOS.TEC.07.013.

[1] http://archive.eso.org/cms/tools-documentation/eso-data-interface-control

[8] Gillies, K., Dunn, J., and Silva, D.R. 2006, Defining common software for the Thirty Meter Telescope, SPIE Conf. 6274, Advanced Software and Control for Astronomy.

[9] Chiozzi, G. et al. 2007, Trends in Software for Large Astronomical Projects, Proceedings of ICALEPCS07, http://accelconf.web.cern.ch/AccelConf/ica07/PAPERS/MOAB03.PDF