Is the workflow model a suitable candidate for an observatory supervisory control infrastructure?

Philip N. Daly[a] and Germán Schumacher[b]

[a]LSST, 950 N. Cherry Avenue, Tucson AZ 85719, USA
[b]LSST, Colina El Pino s/n, Casilla 603, La Serena, Chile

ABSTRACT

This paper reports on the early investigation of using the workflow model for observatory infrastructure software. We researched several workflow engines and identified 3 for further, detailed, study: *Bonita BPM*, *Activiti* and *Taverna*. We discuss the business process model and how it relates to observatory operations and identify a pathfinder exercise to further evaluate the applicability of these paradigms.

Keywords: LSST OCS Control Software Business Process Model BPM Bonita Activiti Taverna

1. INTRODUCTION

The Large Synoptic Survey Telescope (LSST[1]) is a new generation telescope being constructed in Chile over the next several years. It will have an 8.4 metre primary mirror and the world's largest digital camera[2] at 3.2 Gigapixels and carry out a 10-year survey of 18,000 degrees^2 of the southern sky every few days. After 10 years of operations, it will have visited every region of the southern sky ∼825 times with a total exposure time of close to 4 hours per region. The observing strategy will open up the time domain in astronomy and will lead to many exciting discoveries including millions of Type Ia supernovae used as dark energy probes, so-called near Earth objects *etc*. The public web site* is open to scientists and non-scientists alike.

LSST is, therefore, a complex system of systems with demanding performance and operations requirements. A major component of the LSST telescope and site software is the observatory control system (OCS) that schedules, coordinates, commands and monitors the observatory. In another paper,[3] we provided a comprehensive overview of all of the meta-components of the OCS and the, *de facto*, loosely coupled interactions between such meta-components and we reflected on a compartmentalized approach, recognizing the underlying modularity of the system, using a well-known scripting language. That approach describes the current baseline, which has been proven on prior initiatives such as the Southern Observatory for Astrophysical Research (SOAR) telescope.

It is possible that these meta-components will be written in different programming languages reflecting the diversity of their essential functionality. The underlying service abstraction layer (SAL[4]) infrastructure software—a cornerstone development of the OCS—will guarantee interoperability in such a heterogeneous environment. We also promulgated, in that paper, current status and plans that fit the timeline of the construction schedule with current staffing levels and budget.

On a parallel track, however, we have been investigating the workflow model *in lieu* of a more traditional scripting approach. A workflow can be thought of as a *graphical representation of a virtual script*. Software to support such systems already exists. In fact, a *Google* search on the term 'graphical workflow systems' shows just how many choices there are! For LSST, however, we require a Linux (only) run-time environment and wish to support the open source (code) model. Refining a search along these criteria rapidly reduces the number of options ... but it is non-zero. We took a rudimentary look at some choices (*Decisions Platform*†, *Joget*‡

Further author information: (Send correspondence to P.N.D.)
P.N.D.: E-mail: pdaly@lsst.org, Telephone: +1 520 318 8438
G.S.: E-mail: gschumacher@lsst.org, Telephone: +56 51 205347
*http://www.lsst.org
†http://www.decisions.com
‡http://www.joget.org

and *Kepler*[§]) and have identified 3 candidates for further, more detailed, study: *Bonita BPM*[¶], *Activiti*[‖] and *Taverna*[**]. Note that this latter system already has some basic astronomical support for virtual observatory services[5] and in data reduction[6] but not on the controls side of data acquisition.

A suitable choice of workflow system may provide a more integrated approach: an important aspect of this integration is that during workflow *execution*, there is *automatic coordination* of operation of the seemingly individual components (elements and activities) that constitute the workflow. Also, workflows provide an easy-to-use way of specifying tasks that have to be performed during a specific sequence of operations. This 'engine' providing the coordination of the workflows is an integral and essential component of their software functionality and maps onto functionality that we would have to provide in the current OCS development over several meta-components. Taking, for example, the *Bonita BPM* software product we can map their 3 components onto functionality required by the OCS:

Bonita UI Designer is a (web-browser) interface designer that allows us to satisfy many of the components on the *Operator-Remote* meta-component and provide consistent web-based interfaces over a range of devices (laptop, desktop, smart phone, tablet *etc*);

Bonita Portal is the operator control center that also overlaps with components from the *Operator-Remote* meta-component and provides `admin` type oversight of the workflows;

Bonita (Run-Time) Engine runs the workflows and so replaces the *Sequencer* meta-component. Taken together, *Bonita Portal* and *Bonita (Run-Time) Engine* are also known as the *Bonita Platform*.

Bonita BPM also provides the *Bonita Studio*: an integrated development environment (IDE) for their product which has many of the functionalities associated with, for example, `IDLE` or `IntelliJ`. *Bonita BPM* supports Java which is one of the programming languages supported by the SAL.

Even though we are now in the construction phase of the project, we still have a modest window of opportunity (3–6 months) to evaluate this approach to leverage the impressive, open-source workflow engines that are freely available. Part of this evaluation would be with respect to the schedule for OCS construction.

2. WHAT IS BUSINESS PROCESS MANAGEMENT?

In conventional control systems, some form of supervisory control and data acquisition (SCADA) paradigm—the generic term used for the conglomeration of hardware, software and procedures used to control and monitor typically industrial processes—is used to perform complex operational activities. New initiatives in this area emphasize open systems technology over proprietary systems and open-source over closed code.

However, for LSST the strictly hardware-oriented control systems are dealt with at the subsystem level with the OCS providing *supervisory control* of the sequences and procedures. Because of this higher-level abstraction of control, we can contemplate a workflow-based system which focuses on the whole, synchronized, data acquisition *process* rather than the intimate details of specific hardware control.

In such a context, business process management can be defined as:

> 'Business process management (BPM) is a systematic approach to making an organization's workflow more effective, more efficient and more capable of adapting to an ever-changing environment. A business process is an activity or set of activities that will accomplish a specific organizational goal.'[††].

[§] http://www.kepler-project.org
[¶] http://www.bonitasoft.com
[‖] http://www.activiti.org
[**] http://taverna.org.uk and http://taverna.incubator.apache.org
[††] http://searchcio.techtarget.com/definition/business-process-management

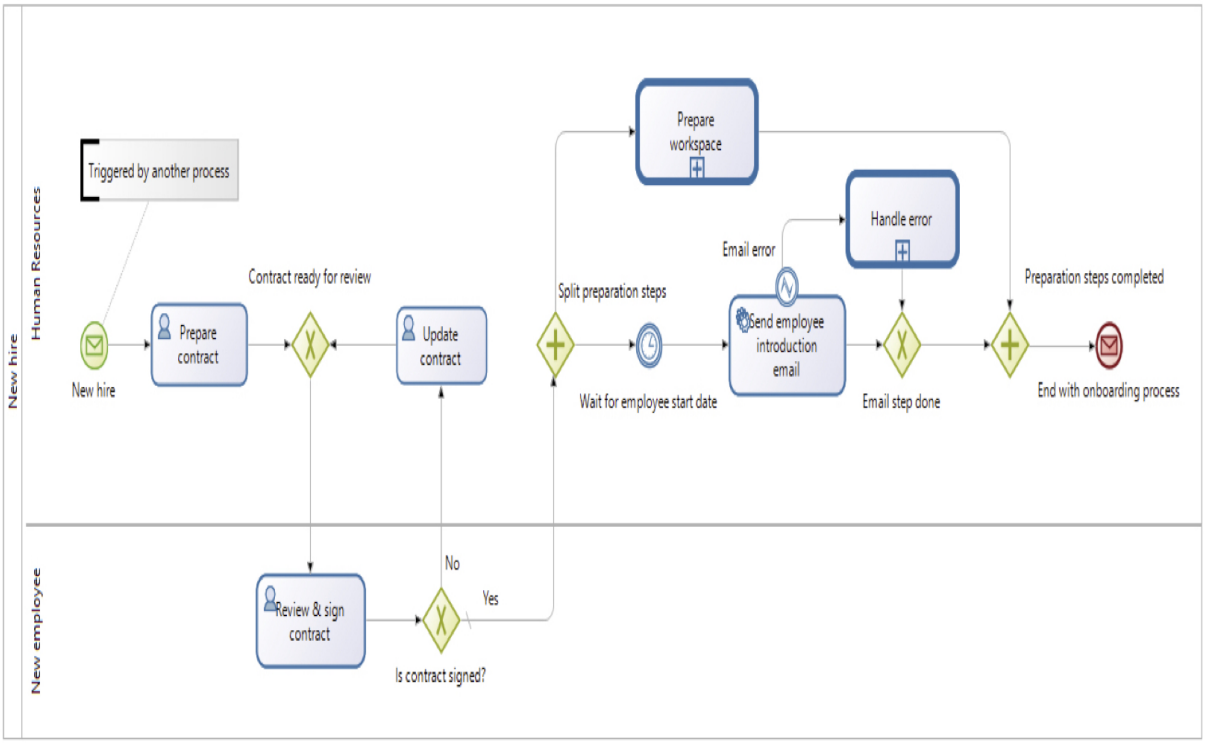

Figure 1. Employee onboarding workflow example. In this workflow, we can clearly identify human-related tasks, automated tasks, error handling, sub-processes, delayed-action timers and multi-task synchronization points.

Business process management is supported by a notation language, BPMN2.0[‡‡], which is an *Object Management Group* (OMG) standard.[7] Indeed, a BPM plugin for PrismTech's *Enterprise Architect* modeling tool (used by the project) is available for integration with the current model if desired but, note, that it does not provide run-time engine support. Also worth noting is that both *Bonita BPM* and *Activiti* support project export and import using XML so, in principle, we have a redundancy built in should one of these organizations cease to exist.

Figure 1 shows a typical business-oriented example of a workflow: one for employee onboarding.[8] Even in this diagram we can see various levels of interaction within the workflow ... some human, some automated, some signals and timers and some error handling. More important is what you do not see within the diagram: the workflow has been developed by an intimate collaboration between end-user (in this case, HR personnel) and the software engineer implementing the back-end of the workflow (model, variables, processes *etc*). A rudimentary approach might be encapsulated in the phrase 'if you can draw the process on a white-board, a workflow can be created for it'.

For the software engineer, BPMN provides the (non-exhaustive) functionality shown in table 1. The usual process for developing a new workflow is:

- Design application pages (web front-end) for 'look and feel' with end user;
- Define the data model;
- Create the process definition (workflow diagrams, variables, contracts, actors etc);
- Define the process forms;

[‡‡] http://www.bpmn.org

Table 1. BPMN Functionality

Tasks	human, service (automated) or call activities (sub-processes)
Events	start, end, messages, timers, error (handlers) and signals
Gateways	parallel (`AND`), exclusive (`XOR`), inclusive
Sequence Flow	conditional flows, default flows
Special Behaviors	looping, nesting, transaction, compensation and correlation
Readability	swim lanes, annotation links
Communications	a variety of communications mechanism are supported
Simulation	simulation using dummy data is built into the workflow engine paradigm

- Use simulated data to debug and test the workflow;
- Deploy the workflow on a suitable platform.

Although workflows can be started from specific event timers (cf. `cron`), it is important to understand that workflows are, inherently, asynchronous by design. They may be started at any time from any legitimate and authorized source. Multiple workflows can be active concurrently. This is, clearly, an attractive idea for LSST in which the main and auxiliary telescopes may be operating independently but—with singular access to shared resources (*e.g.,* the camera, the telescope) also required—a suitable exclusive access mechanism would have to be provided for some use-cases.

Further, it can be argued that the workflow system is best leveraged throughout the observatory and not just in specific applications. Consider the OCS *Maintenance* component: a workflow would be ideal for querying a database periodically to discover the cycle-count on a specific part and automatically create, and schedule, the purchase requisition when 80% of the MTBF has elapsed: a quintessentially Kanban approach.

3. ADVANTAGES

In figure 2, we show a sequence diagram, from the current LSST model, of a standard visit with a filter change. This sequence diagram shows both commands and events associated with the minutiae of control for performing a single use-case. The important aspect of this diagram is that it is *static* (*i.e.,* not executable) but used to verify the model and provide documentation on what the software engineer needs to enable.

Figure 3, shows a workflow of a similar operational sequence. Here, the swim lanes map to principal subsystems of LSST but the transition between such lanes is entirely transparent. The workflow engine provides the synchronization between different subsystems *e.g.,* at 'Gateway1' which waits for both the data management control system (DMCS) and camera control system (CCS) to return. Details of the events produced during this sequence are *encapsulated* and not shown within the activity boxes ('Set Rotator to 0 Degrees', 'Change Filter', 'Move Telescope To Target' *etc*). This produces a clarity regarding the scientific or engineering objective of the workflow. Dummy data can be provided to the workflow for simulation and to debug the data model without impacting any ongoing operations.

Moreover, note that the workflow diagram contains *two* use-cases simultaneously—a filter change and no filter change—and is *executable* in the (real-time) *Bonita BPM Engine*. For the astronomer, the web interface shown in figure 4 provides all the relevant inputs and the workflow would be initiated by clicking on the `Submit` button.

4. WEAKNESSES

Certainly, using a workflow system—*prima facie*—has potential but there are concerns regarding the, naturally simplified, examples and real world use-cases (which may require special activities) and how that impacts the end

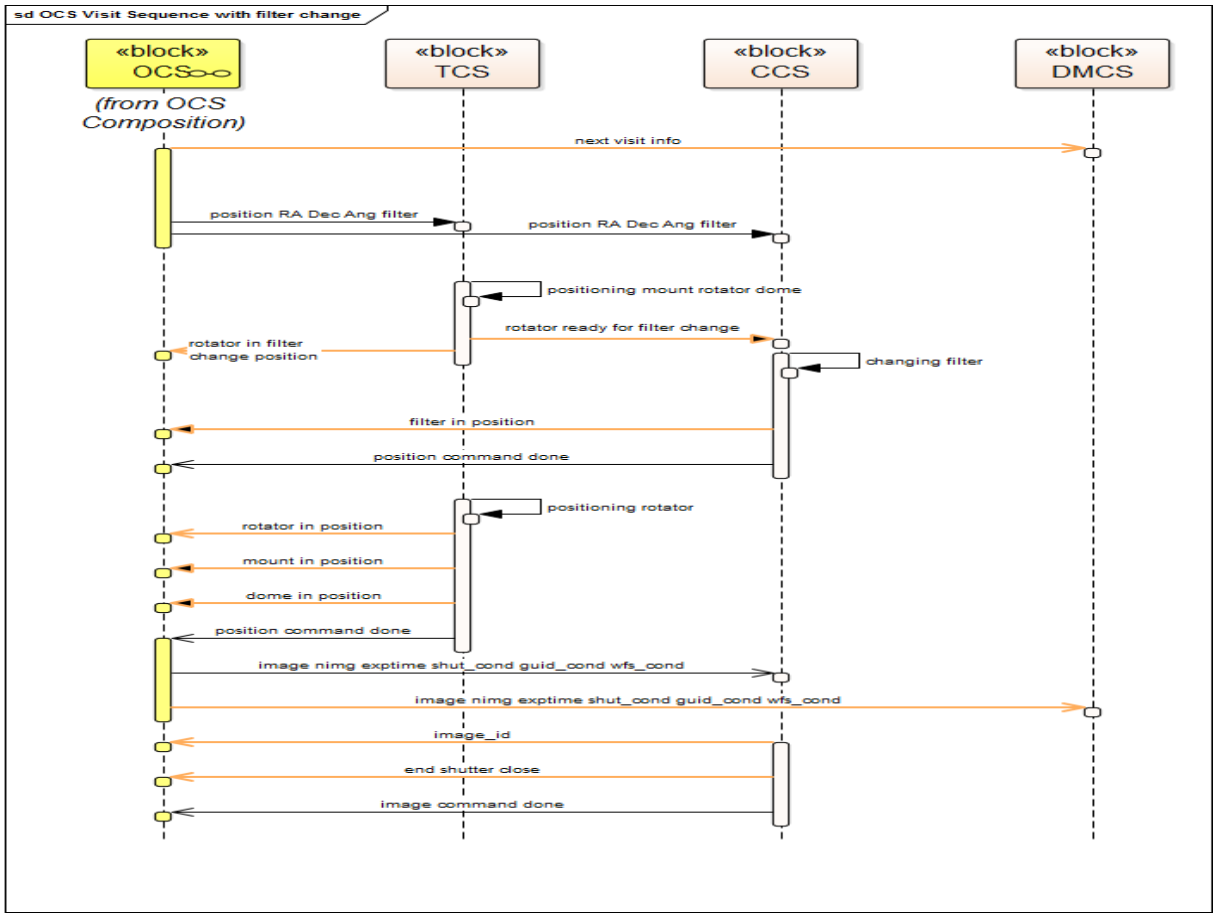

Figure 2. *Enterprise Architect* sequence diagram for a standard visit with a filter change. In this diagram we see command and event flows across the subsystems. The data flows that map onto the figure 3 workflow diagram are *'next visit info'* and *'image nimg exptime shut_cond guid_cond wfs_cond'*.

result. We note that the BPMN standard covers ∼500 pages and 98 elements suggesting a lot more complexity has been found in practice. If a large number of these elements are required to perform modest tasks in observatory operations, the learning curve for new staff might be considerable.

There is a more specific concern when it comes to the requirement on the OCS layer not to add overhead to the observing paradigm. This implies that we might have to look at using persistent objects to represent the camera, telescope *etc* and ways to communicate with these objects would have to be developed.

A further concern is that we may be on the bleeding-edge of new technology for astronomical use. Certainly, the authors know of no other telescope that is considering this approach for controls.

5. PATHFINDER EXERCISE

The next steps for our evaluation are:

1. Connect the workflow system to the SAL layer (probably using a custom connector) and to evaluate the handling of commands, telemetry and events in and out of a workflow;

2. Evaluate the overhead with persistent and dynamic objects;

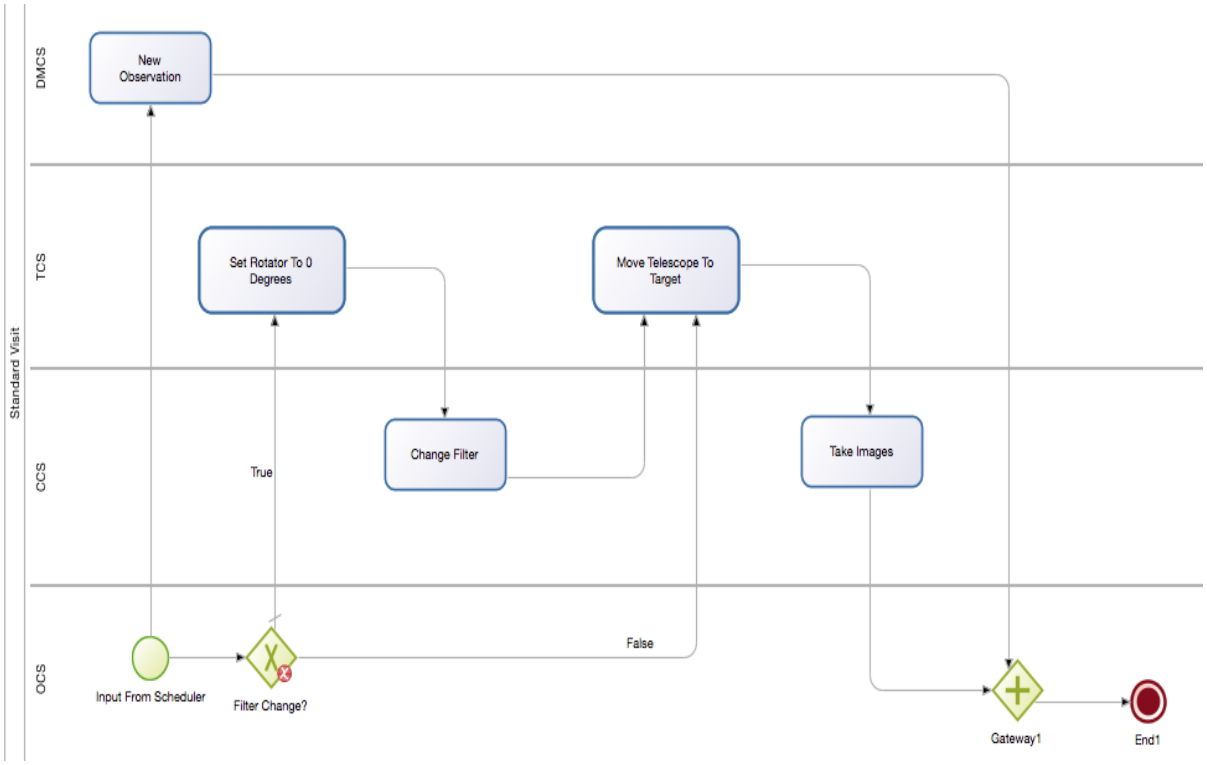

Figure 3. Workflow diagram for a standard visit with and without a filter change in *Bonita Studio*. In this diagram, the essence of the workflow is captured with detailed commands, telemetry and events being subsumed by the activities allowing greater clarity regarding the use of the workflow.

3. Model a typical workflow connected with the observing strategy;

4. Export a use-case model out of one system (using XML) and import it into another to evaluate transferability between different products without loss of work;

5. Evaluate non-BPMN engines such as *Taverna* which is an *Apache Incubator* product.

6. Evaluate the construction timeline implications of adopting this approach.

6. CONCLUSION

Given an *a priori* observing strategy for running the survey, part of what we would want to achieve is to automate implementing that as much as possible, whilst documenting it properly, and this paradigm would seem capable of doing it (whereas a scripting solution would struggle with the documentation). So, if the above pathfinder exercise is successful, the workflow system would be a serious candidate for an observatory supervisory control infrastructure.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation through Cooperative Agreement 1258333 managed by the Association of Universities for Research in Astronomy (AURA), and the

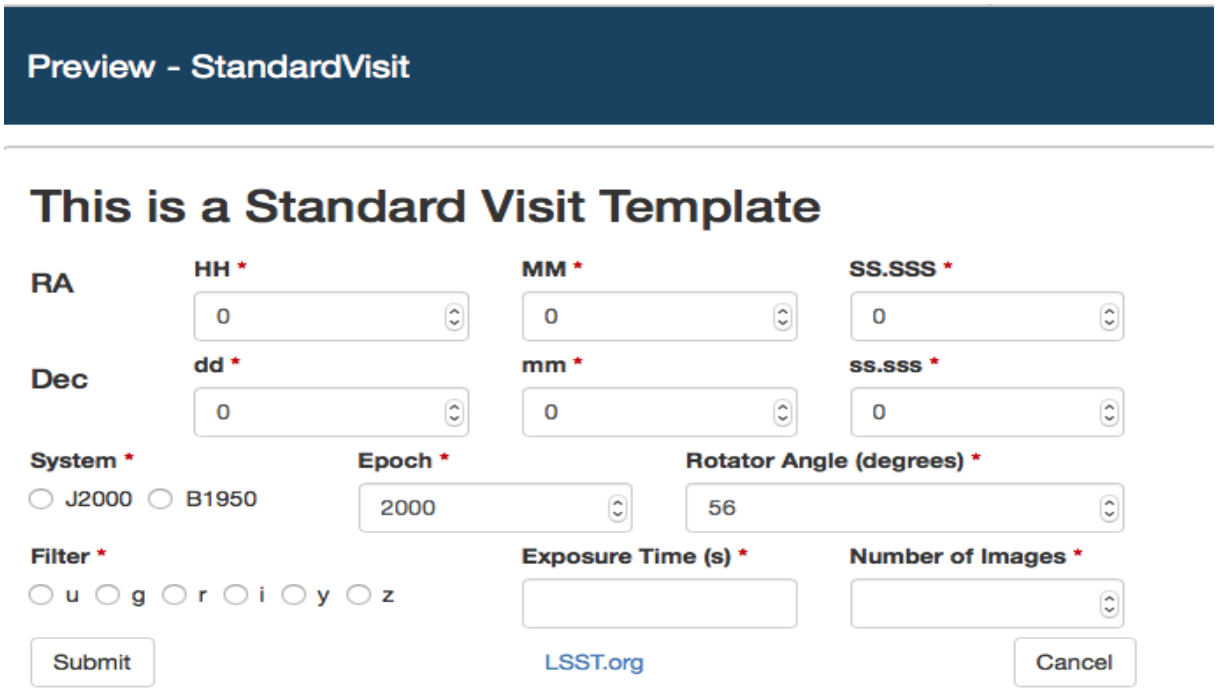

Figure 4. Web interface for a workflow standard visit using *Bonita UI Designer*. This UI is developed using purely web-driven technologies and the data presented to the workflow by the run-time engine when the workflow is started. This is, in effect, the front end to the workflow of figure 3.

Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory. Additional LSST funding comes from private donations, grants to universities, and in-kind support from LSSTC institutional members.

REFERENCES

[1] Kahn, S., "Final Design of the Large Synoptic Survey Telescope," in [*Ground-based and Airborne Telescopes VI*], Hall, H. J., Gilmozzi, R., and Marshall, H. K., eds., *Proc. SPIE* **9906**, in press (2016).

[2] Kurita, N. et al., "Large Synoptic Survey Telescope camera design and construction," in [*Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation*], Navarro, R. and Burge, J. H., eds., *Proc. SPIE* **9912**, in press (2016).

[3] Daly, P. N. and Schumacher, G., "LSST OCS status and plans," in [*Software and Cyberinfrastructure for Astronomy*], Chiozzi, G. and Guzman, J. C., eds., *Proc. SPIE* **9913**, in press (2016).

[4] Mills, D., "LSST communications middleware implementation," in [*Ground-based and Airborne Telescopes VI*], Hall, H. J., Gilmozzi, R., and Marshall, H. K., eds., *Proc. SPIE* **9906**, in press (2016).

[5] Ruiz, J. E. et al., "AstroTaverna-Building workflows with Virtual Observatory services," *Astronomy and Computing* **7**, 3–11 (2014).

[6] Hook, R. et al., "ESO Reflex: A Graphical Workflow Engine for Astronomical Data Reduction," *ESO Messenger* **131**, 42–44 (2008).

[7] "Business Process Model and Notation," *Object Management Group* (2011). http://www.omg.org/spec/BPMN/2.0.

[8] Faura, M. V. et al., "The Ultimate Guide to BPMN2," *BPM Library eBook* **280116**, 24 (2016). http://www.bonitasoft.com/bpm-library.