

# Support tools for the VLT operations: the NTT prototyping experience

Alberto Maurizio Chavan<sup>a</sup>, Gino Giannone<sup>b</sup>, David Silva<sup>a</sup>, Tony Krueger<sup>c</sup> and Glenn E. Miller<sup>c</sup>

<sup>a</sup> European Southern Observatory, D-85748 Garching, Germany

<sup>b</sup>SERCO GmbH, c/o ESO

<sup>c</sup>Space Telescope Science Institute, Baltimore, MD 21218, USA

## ABSTRACT

One of the most important design goals of the ESO Very Large Telescope is efficiency of operations, to maximize the scientific productivity of the observatory. “Service mode” observations will take up a significant fraction of the VLT’s time, with the goal of matching the best observing conditions to the most demanding scientific programmes. Such an operational scheme requires extensive computer support in the area of observation preparation and execution. In this paper we present some of the software tools developed at ESO to support VLT observers, both staff and external. Our Phase II Proposal Preparation system and the Operational Toolkit are prototype implementations of the final VLT systems and have been in use for over a year, while the Scheduling tool, based on the Hubble Space Telescope scheduler and integrated with the Toolkit, underwent field tests at ESO’s New Technology Telescope. We report on our own and our users’ experience in the area of observation preparation, and the application of software scheduling tools to support “service mode” operations.

**Keywords:** Proposal preparation, scheduling, VLT, service observing

## 1. INTRODUCTION

The Data Flow System<sup>1</sup> (DFS) of ESO’s Very Large Telescope (VLT) provides a unified frame for observation data transfer and processing, from Phase I proposal preparation to the final archiving of quality-controlled science and engineering data (see also the papers by Quinn, Silva, Albrecht and Ballester in these proceedings). This paper presents some of the software tools developed within the Observation Handling Subsystem (OHS) of the DFS, dealing with the preparation, storage, scheduling and execution of Observation Blocks<sup>2</sup> (OBs).

Operating a modern telescope has become an exercise in abstraction. Traditional telescope user interfaces evolved from VT100-style, one-screen/one-exposure screens to point-and-click Graphical User Interfaces (GUIs) and command scripts. The observer moved away in time and space: from the eyepiece to the control room, to a remote control station, to his own office.

The DFS has introduced two more levels of abstraction. The Observing Template is a representation of a “typical” instrument operation, designed to provide a high-level, functional view of an instrument, so that the astronomer can concentrate on what to do (*e.g.*: take a I-band image) rather than how to do it (configure the light path, rotate the filter wheel, open the shutter, expose, readout, etc.). The Observation Block combines one or more Templates with target information, defining a modular representation of a self-consistent astronomical observation, like “an Echelle spectrum of star XYZ, together with the relevant wavelength calibration frames”.

While not entirely new, these concepts have now become *the* way to program an 8-meter class, ground-based telescope; the same OB can be executed in service mode as well as by the visiting astronomer, and VLT data reduction and archiving operate by associating data frames to the original OB.

The VLT facility shall be offered in two modes: one with the astronomer present at the telescope, directing the activities of the observatory staff (Visitor Mode or VM), and one where observations are acquired by observatory staff on the basis of pre-determined sequences of operations (Service Mode or SM). ESO is currently planning to devote up to 50% of all VLT time to Service Mode programmes. The OHS has been designed to support both operational modes.

---

Alberto Maurizio Chavan: E-mail: [amchavan@eso.org](mailto:amchavan@eso.org)

Having to plan for service and visitor mode support meant developing a host of software tools capable of dealing with a potentially large (several hundred) pool of heterogeneous observations, as well as other tools which would handle a more limited set of OBs, but in a very flexible way. These different tools needed to handle the same basic concepts (OBs, communication with the VLT Control System), and sport a similar user interface, while supporting radically different ways of operating a large telescope. An iterative style of software development was chosen, together with a flexible development environment (based on interpreted programming languages like Tcl/Tk, Lisp, and SQL). These choices made it possible to react quickly to new requirements originating from early users, and follow a fast release cycle: manpower crises notwithstanding, we could normally give our users the required new capabilities in a matter of weeks, sometimes days.

The need for prototyping was all the more needed in the area of scheduling. To our knowledge, this is the first time that a ground-based observatory attempts to use a software tool to schedule observing queues, and a large number of policy, procedural and technical issues had to be defined even before they could be solved. ESO has now over a year of experience with service observing, during which the scheduling problem itself could be better defined with the help of early versions of the tools.

The OHS tools are used in day-to-day operations at ESO's New Technology Telescope (NTT) in La Silla, Chile, and user feedback greatly helped in shaping the tools' user interface and set of functionalities. Moreover, the tools helped redefine some important operational issues, particularly in the area of service mode (queue mode) observing, a relatively new concept for ESO.

Collaboration between ESO and the STScI enabled us to make use of the experience gained in scheduling the Hubble Space Telescope (HST), although ground-based and space-based telescopes operate in very different environments, and need to take different parameters into consideration. For one thing, while the HST enjoys stable observing conditions, the VLT needs to deal with a variable environment; thus requiring a greater degree of flexibility to react to changing weather.

While the OHS prototypes will be used for the early VLT operation cycles, a baseline version of these tools, incorporating the experience gained so far, will be later deployed.

## 2. THE FRONT-END OF THE VLT DATA FLOW SYSTEM

### 2.1. The Service Mode Perspective

Using an industrial metaphor, one can view the Observation Handling subsystem of the DFS as the OB manufacturing, storage, and shipping and tracking plant. A large fraction of an OB's life cycle takes place within this subsystem.

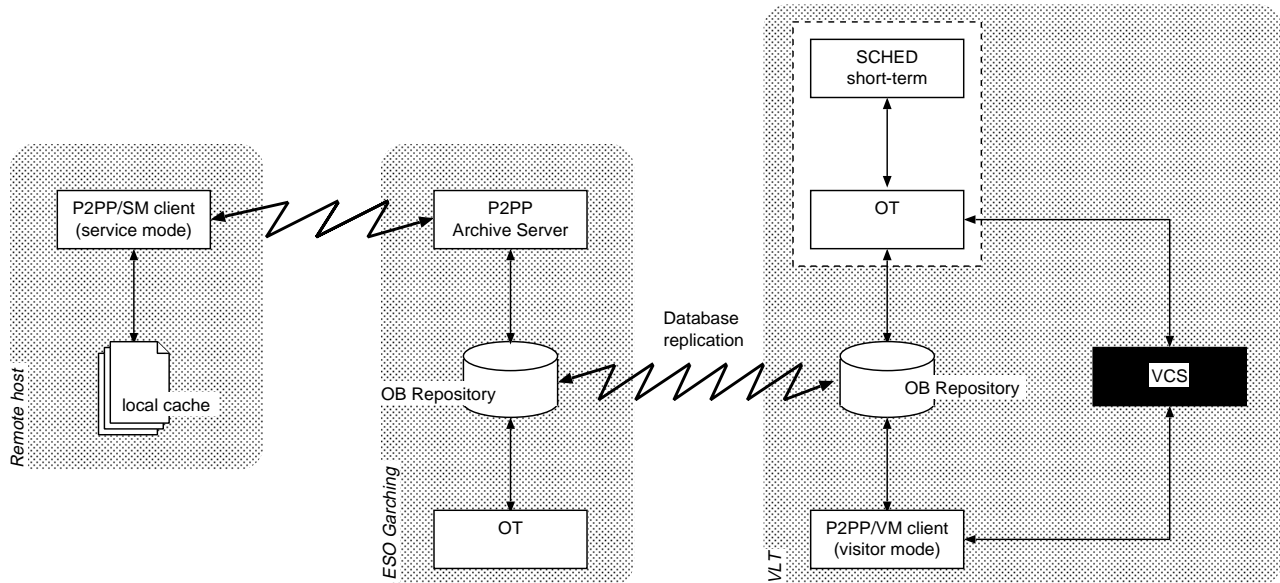
*Manufacturing* of the OBs is performed via the Phase II Proposal Preparation system (P2PP), a collection of distributed processes operating in a client/server fashion. The tool's client side runs on the astronomer's workstation, and enables the user to build and interconnect an OB's components – target description, acquisition strategy, and technical setup of the instrument. Observing Templates (more simply, Templates) provide the user with a high-level view of the instrument (see also sec. 1).

*Storage* of OBs can be done on the local workstation (in the so-called *local cache*), but OBs must be transmitted to the central ESO repository before they can be verified and executed. P2PP's *Archive Server* runs at ESO's headquarters in Garching, providing storage and security services to the community of users. Physical storage and repository administration is handled via a commercial relational DBMS. The repository is replicated to the VLT, so that both the observatory and the headquarters share up-to-date information.

Stored and verified OBs are then *shipped* to the VLT Control System (VCS) for execution. Queuing OBs for execution and scheduling the queued OBs during a specific night are performed by the Operational Toolkit (OT) and Short-Term Scheduler (SCHED) systems.

As an OB progresses through the VCS and the back-end subsystems of the DFS (Archive, Pipeline and Quality Control), its status is *tracked* and archived in the repository. In fact, as an OB is executed, archived, and reviewed during the quality control process, more and more information is associated to it, in terms of both a history of events and the actual scientific and engineering data it produces.

The software architecture of the DFS front-end is shown in Fig. 1. Its different components run at the astronomer's home institution, at ESO's headquarters in Garching, and at the VLT, where they interact with the VCS (shown symbolically as a "black box").



**Figure 1.** Software architecture of the DFS front-end.

Note that P2PP/VM indicates “visitor mode” P2PP (see sec. 2.2), while P2PP/SM stands for “service mode” P2PP.

## 2.2. The Visitor Mode Perspective

Visiting astronomers enjoy a greater flexibility than astronomers relying on ESO’s observing services. Their OBs do not need to be prepared in advance, and last minute changes are always possible — while, as a rule, staff observers do not modify scheduled, service mode OBs.

Moreover, visiting astronomers generally have to deal with a limited number of OBs, or OB types; and know their observing programme intimately. For them, building and scheduling queues can be an added burden rather than an benefit.

While the combination of P2PP, OT, and SCHED can be used by visiting astronomers as well as staff observers, it is often more convenient for visitors to use a special version of P2PP, called *P2PP/VM* (from “visitor mode”). P2PP/VM combines manufacturing, storing, shipping and tracking of OBs in a stand-alone software tool; the observer can create and edit OBs on the fly, and use OB execution feedback to adapt to the current observing conditions and results.

With P2PP/VM, OB storage, shipping and tracking is handled in a transparent way, so that the observer need only be concerned with their scientific goals. The VLT is still operated fully within the DFS framework, and can provide the complete range of back-end DFS services (archiving, on-line pipeline processing, etc.).

As shown in Fig. 1, P2PP/VM has its direct connection with the VCS and the OB repository, allowing it to be stand-alone. Its user interface is very similar to that of P2PP/SM (from “service mode”), so that experienced users can operate with either version using the same set of familiar widgets and menus. P2PP/VM, however, does not allow the user to define scheduling constraints (see sec. 3.2.2), nor to interact with the P2PP archive server; while P2PP/SM cannot interface directly to the VCS.

### 2.2.1. P2PP/VM Usage Reports

Early users of P2PP/VM found the tool and the underlying operational concept “straightforward” and “flexible”. In general, they favored the idea of preparing OBs before coming to the observatory, so that “you don’t have to think about everything at four a.m.”, and felt that a few hours of hands-on training were enough to be fully operational. Their overall rating ranged from “good” to “very good”; experienced users found P2PP/VM “an improvement” over the pre-DFS control software of the NTT.

The prototype received quite some criticism, too. The user interface was defined too crowded, and lacking “visual clues”: all information is presented in the same way, regardless of its relative importance. Some functionalities are missing, or are not easy to find; some operations require too many mouse clicks; and of course, performance could be improved.

Although only half a dozen visitors used P2PP/VM to observe with the NTT, their reactions are encouraging and stimulating. This kind of user feedback is extremely important; in fact, it helped defining the evolution of P2PP ever since its first in-house users.

### 3. THE OPERATIONAL TOOLKIT AND SHORT-TERM SCHEDULER

The Operational Toolkit and the Short-Term Scheduler are two subsystems of the same integrated software tool, sharing a common GUI. The OT is used to create candidate lists of Observation Blocks (*queues*) to be observed during a “service observing run” — usually, a few nights (3 to 7). SCHED takes these queues of observation blocks as input to create timelines, assigning each Observation Block an expected execution start time.

#### 3.1. Operational scenarios

Before we describe the structure of the Short-Term Scheduler, we give in this section an operational overview of the OT/SCHED combination.

##### 3.1.1. Preparing an Observing Queue with the OT

The main components of the Operational Toolkit are the Repository Browser and the Queue Manager. The Repository Browser can be used to browse the ESO Observation Block repository, and select and sort OBs according to different criteria, including RA and Dec, observing programme, moon and weather constraints, and so on; selected OBs can then be appended to a queue.

One or more alternative queues can be created per each service observing run (queues should in general be oversubscribed, in order to cover different weather conditions and provide more flexibility when building timelines). The Queue Manager is the environment in which the user can edit queues, reorder OBs, inspect them with a graphical tree analyzer, and produce queue reports.

##### 3.1.2. Preparing a Medium-Term Schedule with SCHED

When preparing a schedule for a queue, the Operational Toolkit and SCHED tools are usually operated as follows.

1. SCHED is run with a scheduling time range encompassing the whole observing run, and some expected weather conditions (“photometric sky, excellent seeing”; or “thin cirrus, medium seeing”; etc.) This operation enables the user to create a medium-term schedule, which covers the whole observing run, with the same schedule resolution as in the short-term schedule (by default, twelve seconds). This operation can help detect scheduling problems, like OB under-subscription in some RA ranges or moonlight illumination conditions.
2. The queue is re-defined with the Operational Toolkit, selecting new OBs with the Repository Browser and removing non-schedulable OBs.
3. Repeat the previous steps until the queue fits the expected range of observing conditions.

##### 3.1.3. Executing a Short-Term Schedule with OT and SCHED

At the beginning of each observing night, SCHED is run again, with a one-night scheduling time range and the current weather conditions as inputs; its output helps the Data Flow Operations (DFO) manager decide on which OBs to execute and when, as well as how to configure the VLT facility for science operations that night.

SCHED is usually run several more times during an observing night, to compensate for unexpected events or varying weather conditions.

OBs selected by the DFO manager are passed to the VLT Control System (VCS) for execution; VCS feedback is received as asynchronous *events* and stored into the the OB repository.

## 3.2. Short-Term Scheduler

Although the Short-Term Scheduler could in principle be run in a completely automatic way during the observing nights, it is really configured as a Decision Support System, helping a user in defining timelines for an observing night — the final decisions belongs to the user, which can choose the most appropriate scheduling criteria for the current situation.

It is well known that scheduling is a highly combinatorial, computationally hard, NP-complete problem; and nightly scheduling of the VLT is no exception. The features of SCHED are best described by listing its goals, the constraints it must respect, and the strategies it employs.

### 3.2.1. Scheduling Goals

The main goal of SCHED is that of maximizing the utilization of the telescope time, but there exists a set of other, potentially conflicting goals.

- give greater preference to OBs with higher scientific priority, as defined by ESO's Observing Programme Committee (OPC) recommendations;
- minimize instrument (and instrument mode) setup time: from a operational point of view this makes more sense then the previous goal, because it implies also an optimization of the calibration plan;
- minimize the airmass at which OBs are executed;
- give higher preference to OBs matching the current weather conditions: if observing conditions are very good, for instance, any OB can in principle be executed, but it makes more sense to take advantage of the unusual situation and execute OBs with more stringent weather requirements.

It is easy to understand how these goals may conflict with each other, and how scheduling algorithms which cover only one of these goals are not really useful in telescope operations.

### 3.2.2. Scheduling Constraints

OBs are subject to different kinds of constraints, either astronomical (*e.g.*, objects can only be observed when they are above the horizon at nighttime), related to the weather (*e.g.*, this is a faint object which can only be observed under very good seeing conditions), or limiting the admissible observing date and time:

- target viewing (an observation can be executed only when the target is visible);
- moon-target angular separation (some OB targets must be a further from the Moon than some minimum angular separation);
- fractional lunar illumination (some OBs can not be executed if moon brightness is above a given threshold);
- required airmass (the Principal Investigator may require that the target be observed only within a given airmass range);
- required sky transparency and seeing;
- absolute timing constraints (the OB has must be executed within one or more pre-defined time intervals);
- relative timing constraints; *i.e.*, sequencing relationships: *e.g.*, “execute OB<sub>2</sub> a week after OB<sub>1</sub>”.

These two lists of goals and constraints are the reason why we implemented new scheduling algorithms within the SPIKE environment.

### 3.2.3. Scheduling Strategies and SPIKE

SPIKE<sup>3</sup> is the scheduler used by the operations team of the Hubble Space Telescope. It was adapted to ground based telescopes requirements by the Space Telescope Science Institute in Baltimore, MD, and is a very good scheduling environment which can be easily used and customized. The basic SPIKE environment includes some generic scheduling algorithms, which were modified to create several new strategies.

By definition, scheduling algorithms respect all scheduling constraints, but they do mix and match the different goals, assigning different relative weights to each; however, all our strategies give greatest weight to the main goal of maximizing the utilization of the telescope time.

Different strategies can be useful, because the user can run the algorithm most appropriate to different situations; or compare the timelines resulting from applying different strategies to the same input data.

The system is intended to support the user, in the sense that schedules can be generated automatically, semi-automatically (with the user interactively locking OBs to a given start time, and then running the scheduling algorithms on the remaining OBs) or completely manually (user chooses a start time for all OBs). In any case, much scheduling information is available to the user, both in textual and in graphical format.

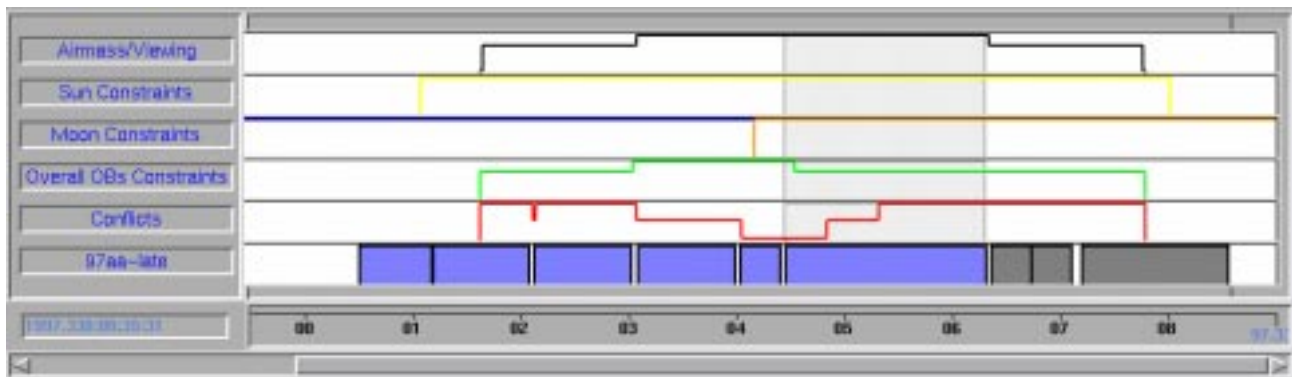
Note that SPIKE includes also a large library of functions to compute Moon, Sun and target positions, relative distances, etc. Using these functions, astronomical constraints (sec. 3.2.2) can be computed and represented graphically (sec. 3.3), providing a synoptic summary of an object's visibility and airmass.

### 3.3. The Graphical User Interface of the Short-Term Scheduler

The graphical user interface (GUI) of the Short-Term Scheduler is divided into two main components: an information area containing textual information, and a graphical representation of the current timeline.

The information area contains the lists of all Observation Blocks of the current queue, and of all schedules computed on the queue itself. Detailed information is available for each computed schedule, and for each scheduled OB.

The horizontal axis of the timeline widget (shown below in Fig. 2) represents time, and each box represents an OB, which can be interactively moved and locked to its new position (start time). A number of histograms are displayed for the currently selected OB, including an approximation of the airmass/viewing preference curve and a profile of the sun and moon constraints (both in terms of set and rise times, and acceptable distance from the target). A final histogram represents the overall constraint set, that is, the intersection of all constraints and preferences, showing when the selected OB can and cannot be executed.



**Figure 2.** The timeline widget of the Short-Term Scheduler Graphical User Interface

### 3.4. NTT Test Results

In keeping with our plans,<sup>4</sup> we installed and tested SCHED at the NTT, during the month of January 1998. The tool was used off-line, scheduling real science observations during a service observing run. Although the test was too limited\* to elicit the kind of user feedback P2PP/VM had (sec. 2.2.1), it was still meaningful, as we could assess the tool's scheduling flexibility and power, with real data samples and under real operating conditions.

Since NTT service observers do not normally prepare written schedules before an observing night, so it was not possible to benchmark SCHED's output against a manually produced document; however, the NTT staff observer acknowledged the usefulness of SCHED in quickly reacting to weather changes or unexpected events — strategies were re-run with updated parameters several times during a night. Often, SCHED highlighted scheduling problems that had gone unnoticed when inspecting the OB queue.

SCHED was also appreciated while defining the observing queue. Medium-Term Schedules were computed for the whole queue, with a scheduling time range equivalent to the duration of the service observing run (6 nights), and helped in re-defining the queue itself with more precision.

Table 1 shows an assessment of the different scheduling strategies of SCHED. Each row of the table shows the results of using a different scheduling strategy; the first five algorithms have been developed specifically for ESO, while the last two are SPIKE built-in algorithms. The ESO strategy code is built from the letters used to represent five different goals:

*E*: maximize telescope time efficiency: that is, minimize schedule gaps; in Table 1, *E2* identifies a strategy which gives an even higher weight to this goal;

*W*: give higher preference to OBs matching expected weather conditions;

*R*: give higher preference to OBs with higher rank<sup>†</sup>;

*A*: minimize airmass;

*I*: minimize instrument changes.

Each column reports the how well the corresponding goal is served by the strategy:

*Time efficiency*: percentage of telescope time efficiency;

*Airmass efficiency*: percentage of airmass efficiency: *i.e.*, how close OBs are to their optimal airmass;

*Rank*: mean priority of the scheduled OBs: lower numbers indicate a better choice;

*Weather efficiency*: percentage of weather efficiency: *i.e.*, how often OBs are executed with the best matching weather conditions;

*Time*: average, minimum and maximum execution times on a standard Unix workstation.

Values in the table are the mean values over ten runs, with different NTT data sets and weather conditions. The scheduling time range for all runs was one night, the number of OBs ranged from 30 to 90 per each run.

**Table 1.** NTT scheduling test results

SCHED goal coverage for different scheduling strategies						
Strategy code	Time eff. (%)	Airmass eff. (%)	Rank	Weather eff. (%)	Execution time	
					avg	min-max
EWRA	85.0	95.3	1.34	74.1	24	8-60
E2WRA	87.5	95.5	1.32	74.3	21	8-57
RWA	84.9	95.1	1.34	72.6	37	10-91
EWRI	86.3	93.5	1.29	73.6	21	8-56
RWI	82.7	94.5	1.38	72.1	38	9-95
Early greedy	94.1	81.8	2.20	67.3	9	5-15
Max-pref	75.5	95.1	1.70	71.0	31	10-90

\*The weather was unusually bad, and we lost several observing nights due to heavy cloud cover.

<sup>†</sup>Observing programmes are ranked by ESO's Observing Programme Committee; OBs have the same rank as the corresponding programme. 1 means highest priority, 3 lowest.

Scheduler run times are acceptable: in the worst case it takes less than 100 seconds to execute one algorithm, but the mean time is 30 seconds (these values should be compared to the time needed by an operator to manually schedule a whole night).

ESO strategies show a good coverage of potentially conflicting goals. The *Early greedy* algorithm yields the best time efficiency, but other values — like rank, airmass and weather efficiency — are unacceptable (of course, it is easier to make a schedule which is time-efficient than one which is time-efficient *and* makes scientific sense); while the *Max-pref* maximizes a single goal (airmass), at the cost of a lower time efficiency.

#### 4. CONCLUSIONS AND PLANNED DEVELOPMENTS

The OHS tools implemented at the NTT helped define some important operational concepts, and provided support to both external and staff astronomers doing real-world science observations. However, work is still needed in the areas of robustness, ease of use and efficiency — some important functionalities are missing, as well. We plan to re-implement the current tools using a compiled language, to compensate for the performance and maintainence deficiencies of the current prototype. Although it is still relatively unstable, we think that Java will be a viable implementation language, offering the added benefit of the promised “write once, run anywhere” feature — our prototypes can currently run only on the Solaris and HP-UX flavors of Unix.

SCHED will continue using the Lisp-based SPIKE core. We plan to use SPIKE also for our upcoming Long-Term Scheduler, which will help scheduling the observatory activities (that is, service and visitor observing runs, and technical time periods) over a six-month time frame.

#### ACKNOWLEDGMENTS

The authors would like to thank Marco Scodreggio and the NTT team in La Silla, lead by Gautier Mathys, for their friendly and collaborative support.

#### REFERENCES

1. M. Peron, M. Albrecht, P. Ballester, K. Banse, A. M. Chavan, P. Grosbol, P. J. Quinn, and D. Silva, “VLT Data Flow System: the NTT experience,” in *Telescope Control Systems II*, H. Lewis, ed., *SPIE Proceedings Series* **3112**, pp. 60–65, 1997.
2. A. M. Chavan and M. Albrecht, “A Distributed System for “Phase II” Proposal Preparation,” in *Astronomical Data Analysis Software and Systems VI*, G. H. . H. E. Payne, ed., *ASP Conference Series* **125**, pp. 367–370, 1997.
3. M. Johnston and G. Miller, “SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations,” in *Intelligent Scheduling*, M. Zweben and M. S. Fox, eds., pp. 391–422, Morgan Kaufmann, San Francisco, 1994.
4. A. M. Chavan, G. Giannone, D. Silva, T. Krueger, and G. Miller, “Nightly Scheduling of ESO’s Very Large Telescope,” in *Astronomical Data Analysis Software and Systems VII*, *ASP Conference Series* **(to be published)**, 1998.