A Versatile Scheduler for Automatic Telescopes

Sabine Moisan[a], Michel Boër[b], Carole Thiebaut[b], Fabien Tricoire[a], and Monique Thonnat[a]

[a]INRIA, Sophia Antipolis, France [b]CESR/CNRS, Toulouse, France

ABSTRACT

We present a versatile scheduler for automated telescope observations and operations. The main objective is to optimize telescope use, while taking alerts (e.g., Gamma-Ray Bursts), weather conditions, and mechanical failures into account. Based on our previous experiment, we propose a two steps approach. First, a daily module develops *plan schemes* during the day that offer several possible scenarii for a night and provide alternatives to handle problems. Secondly, a nightly module uses a reactive technique –driven by events from different sensors– to select at any moment the "best" block of observations to launch from the current plan scheme. In addition to a classical scheduling problem under resource constraints, we also want to provide dynamic reconfiguration facilities. The proposed approach is general enough to be applied to any other type of telescope, provided that reactivity is important.

Keywords: robotic observatory, scheduling, state-transition model, reactive systems

1. INTRODUCTION

Most of the recently built ground-based telescopes feature automation of the planning and scheduling tasks, either to assist the service astronomer or in the framework of a fully automated observatory. The importance of efficient scheduling is increasing since telescopes and their instrumentation are becoming more versatile and complex, and they produce more data. Moreover, the tasks of image acquisition for a given project are often delegated to a "service astronomer" or to a software in the case of fully automated observatories.

In this latter context, it is necessary to use software tools that optimize the scientific return of the instrument with respect to its global science project. It is possible to summarize the goal of such tools as follows: to observe as many as possible sources, each in "optimal" conditions, given a set of criteria (e.g. priority, constraints...), taking into account that observations may be interrupted at any time by bad weather or by a high priority Target Of Opportunity (TOO, e.g., the occurrence of a cosmic Gamma-Ray Burst). In addition, the observing conditions may become "superb" (according to the appropriate criteria), hence it may be more desirable to schedule requests that necessitate such conditions. In the normal (man-operated) observatory context, the user is given a certain amount of observing time (hours, nights...) which may be eventually performed in "service mode", i.e. by the observatory staff.

A mixed approach has been introduced for the ESO VLT and NTT[1] with the concept of "observation blocks", defined as units of programming. Each accepted proposal is described in terms of observation blocks which are then handled by three schedulers respectively in charge of the long term, the medium term, and the short term time line, with a degree of reactivity growing from the long to the short term. The complexity of the system comes primarily from the need to cope with observations performed by guest as well as service observers and from the multi-telescope nature of the VLT. In a somewhat different approach, Bresina[2] defines individual groups of observations sent to the telescope via a centralized system, the "Principal Astronomer", with pre-allocated priorities and time constraints. The system chooses the best sequence which optimizes an objective function. However, TOO observations as well as changes in the schedule are hardly taken into account by this system.

In this paper we describe the tools we are developing to build an automatic versatile scheduler in the context of the ARAGO (Advanced Robotic Agile Guest Observatory[3]).

(Send correspondence to Sabine Moisan) S. Moisan: E-mail: Sabine.Moisan@inria.fr
M. Boër: E-mail: Michel.Boer@cesr.fr

We propose a two steps approach:

- A daily module develops forecast *plan schemes* during the day. These plan schemes dispatch the requested blocks of observations over several nights (with a variable limit, e.g., one year). Plan schemes offer several possible scenarii for a night. In particular, since computation time is not limited, this module can foresee as many problems as possible that may occur during the next night (bad weather, resource shortage,..) and provide alternatives to handle these problems.
- A nightly module then dynamically schedules feasible blocks of observations, depending on the evolving situation. It receives events from different sensors and reacts by selecting the "best" block of observations to launch from the current plan scheme. The plan scheme provides this module with a pre-processed reduced search space, avoiding unnecessary computations during the night. The main objective is to optimize telescope use, while taking alerts, weather conditions, and mechanical failures into account.

In the next section (2) we describe shortly the ARAGO project and the particular context of scheduling for this telescope. After a brief survey of our previous tool (section 3) designed for the autonomous TAROT instrument, we detail in section 4 our specification of a new automatic scheduler for the ARAGO telescope. Finally, we discuss the generality of the described framework and we present future work.

2. THE ARAGO PROJECT

The goal of the Advanced Robotic Agile Guest Observatory (ARAGO) is a ground-based survey of the variability over the sky in the range from less than one second to several months - year. Among its scientific objectives[3] are the detection of Gamma-Ray burst optical counterparts, orphan afterglows, extrasolar planets, large Kuiper objects, etc. ARAGO will be a fully autonomous observatory featuring a 1.5m telescope and a 10K x 10K CCD camera with a field of view of 2 degrees. The main characteristics of ARAGO are summarized table 1.

Table 1. Main technical characteristics of ARAGO

Aperture	1.50 m
Field of view	2 deg $\times$2 deg
Angular resolution	0.7arcsec
Mount type	Alt $-$ azimuthal
Axis speed	adjustable, up to 80 deg /s
CCD camera size	10k $\times$ 10k pixels
Pixel size	15 μm
CCD readout noise	$\approx 8\,e^-$
Readout time	1 s
Filter wheel	6 pos. : Clear, U, B, V, R, I

The main peculiarity of ARAGO –as well as of TAROT[4,5]– of relevance for the scheduling of telescopes, is its reactivity to any un-programmed event and the absence of people in charge of the instrument, whenever at night or day. On the other side, contrary to other telescopes, ARAGO will have only one instrument at focus, the above mentioned 10K x 10K CCD camera, with its shutter and a 6 filter wheel. In other words there is no change in focal instrumentation.

In this framework, our objective is to design a versatile scheduler for night observations and operations. The scheduler is the intermediary between the users, the telescope itself, and the image processing system. Requests may come from different users and even from other systems: an astronomer, a lecturer, or an observation device (like a satellite, e.g. HETE or SWIFT) may submit a request to the telescope via the Internet. A request is usually composed of several observations, each one indicating the source coordinates in the sky and information about duration, filter to use, periodicity, etc. Other kinds of requests concern the telescope maintenance

(typically calibration operations) that should happen regularly to ensure a correct quality of service. Finally, alerts (such as occurrences of Gamma-Ray Bursts or detections of extra solar planets) may interrupt the current processing at any moment. Every request is assigned a priority: alerts have the highest priority, other requests must respect their user's priority quotas.

Hence, the job of our observation scheduler is first to establish a preliminary optimized observation agenda over about one year, taking into account the various constraints. The goal is to satisfy as many requests as possible all over the year, trying to minimize useless periods: For instance avoid unnecessary filter changes, telescope moves, etc. by grouping observations sharing the same characteristics.

Secondly, it should also adapt its process to dynamically take into account alerts (mainly occurrence of Gamma-Ray Bursts) and a wide range of disruptive events (bad weather conditions, engine breakdown, power failure...). Thus, in addition to a classical scheduling problem under resource constraints, we also want to provide dynamic facilities to adapt the schedule of the current night and of the following ones. This means to introduce new requests raised by alerts, to shift requests that cannot be executed due to problems, and to resume the execution of interrupted observations when possible.

3. PREVIOUS EXPERIENCE: THE MAJORDOME

We have already developed a method for the scheduling of astronomical automatic telescopes, in the framework of the autonomous TAROT[6] instrument. The MAJORDOME[4] software can handle a variety of observations, constrained, periodic, etc., and produces a time line for the night, which may be modified at any time to take into account the specific conditions of the night. The MAJORDOME can also handle target of opportunity (TOO) observations without delay.

In this paper we use the same definitions as presented in section 3 of the previous MAJORDOME paper.[4] We summarize below the main terms and concepts introduced in this paper:

- A *request* is a set of observations requested by a distant user, including their description, constraints, periodicity, etc.
- An *observation block* (or *block*) is a set of contiguous observations scheduled together. Hence a *block* is the validation and programming unit for the MAJORDOME.
- Any user can send a request at any time, the lifetime being set by the user, or the observatory team (currently a maximum of 1 year).
- The user can set one or more constraints (period, date ...), but usually does not know when the request will be scheduled.
- All the scheduled requests (apart other constraints) should be observed in the best possible conditions (e.g. minimum airmass, moon phase...).
- In the following the terms of *night window* and *observation window* refer to their definitions in Bringer et al.[4]

In the case of MAJORDOME, the time line is computed every day for the next night, using all the visible requests from the database. As soon as a block has been validated, the corresponding request is removed from the database, if not periodic. This software schedules observations with an efficiency above 90%.

However, we encountered some difficulties when handling periodic or constrained requests with a large time interval, and MAJORDOME is unable to give to the telescope users and operators the visibility over the schedule. We also believe that a larger programming horizon would result in more efficient optimization. Moreover, in this version of MAJORDOME, if an unexpected event happens (TOO, interruption...), the current schedule is aborted and a new one is computed for the remaining of the night to address the alert observations. The normal process resumes the day after, including the un-observed scheduled requests.

4. NEW PROPOSAL: MAJORDOME-II

The experiment with MAJORDOME allows us to refine the specifications and to derive a next generation tool. We present the initial specification of our automatic scheduler for the ARAGO telescope. This specification is intended to overcome the limitations of the previous MAJORDOME.

Evben if most of the concepts which ground MAJORDOME are still suitable and should be kept (e.g. observation blocks[1] which are widely adopted), MAJORDOME-II should offer new facilities such as planning long-term observations, simply modifying scheduling strategy, and accommodating new functionalities. It must also be able to react to uncertainties and unexpected events that may occur during execution, with the fewest modifications of pre-planned observations. The current MAJORDOME version rules out forthcoming observations when a GRB occurs, to concentrate only on the new observations generated by the burst. Contrary to this version, the idea for this work is to keep as many pre-planned observations as possible.

Our software development focuses on two fundamental points: (i) efficiency in telescope use (i.e., both maximizing scientific return and minimizing waste of time) (ii) producing a modular, extensible, easy to tune implementation (i.e. the tools should be easy to modify and tune for various needs).

The proposed system is divided into two main modules which are described in the following sections.

4.1. During the Day

Since the requests may arrive at any time (e.g., via Web forms filled out by astronomers), the first task is to automatically sort the requests and to rejects unfeasible ones (part of the sky not visible from telescope, overload of periodic requests, etc.).

Once this first task done, the daily module of MAJORDOME-II aims at producing plans of quality, by planning remaining observations under optimal conditions, like minimal airmass and low background light. It should also juggle various constraints over a limited time. Astronomical constraints of observing requests concern e.g., orientation, observing windows, target visibility, required Moon phase, links among different observations (especially periodicity), and critical times. Resource constraints concern for instance, the availability of filters, as well as time quotas, or sky zone occultations. This module also applies a "fairness" and scientific priority policy to ensure an equitable distribution of observing time among users (i.e. scientific campaigns/programs). Hence, the plans are optimized with respect to an arbitrary combination of many criteria and a trade off must be identified. This trade off can be modified by astronomers depending on their objectives.

These considerations are common to most planning and scheduling tools. Apart from trade off modification by astronomers, our main originality lies in the result of this daily module, which does not generate one "best" plan but a set of possible good scenarios for a night, named an observation *plan scheme*. Such a structure incorporates alternatives to anticipate usual perturbations, such as changing weather conditions or mechanical failures.

A plan scheme can be seen as a reduced optimized search space. It is based on an arbitrary division of the sky into a small number of sectors (around ten). A sector is an area of the local sky (in horizontal coordinates)*. The daily module populates these sectors along the night with blocks of observations with the highest possible priority. The objective is to obtain a balanced distribution of blocks of observations in all sectors. "Balanced" here means that blocks of observations are dispatched into all sectors in an equitable manner. No sector should be empty at any time. In one sector only the best block(s) of observations is(are) selected. Since periodic observations are the most complicated to achieve, they are scheduled first and are given a high priority. Tracking observations that cover several sectors during the night are also scheduled right from the start. These two kinds of time consuming observations have a limited quota in a night (for instance no more than 10% of the total observations).

A plan scheme thus presents a spatial repartition of scientifically interesting observations, with no presupposition on the ones that will be actually performed. A plan scheme may thus lead to many potential lines of

*We also add a pseudo sector, corresponding to calibration observations. It may be used as spare solution, for instance in case of bad weather when any other observation is impossible.

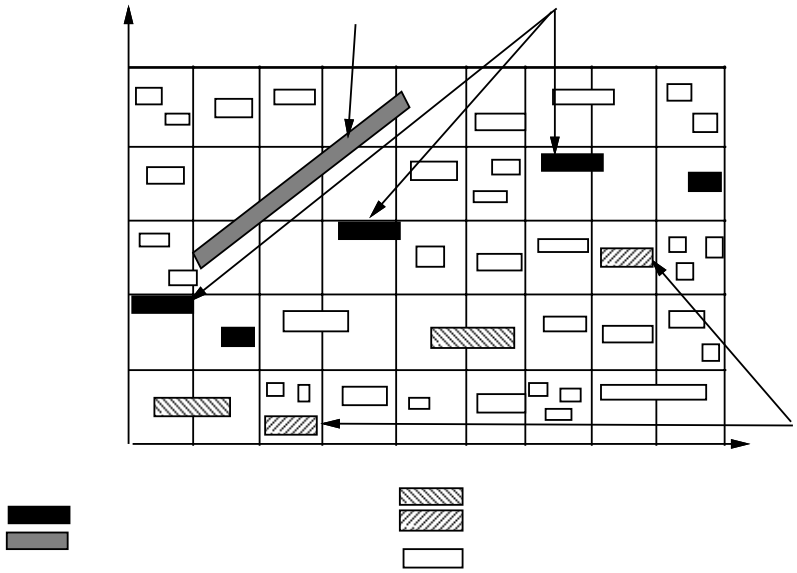


Figure 1. A simplified plan scheme with 5 sectors and regular division of time during the night (under the hypothesis that blocks do not cover more than one sector, except tracking). The block labeled TO is the observation of an object moving across the sky.

execution. In addition to the division into sectors, interesting relations corresponding to other constraints than visibility are maintained among blocks in a plan scheme (for instance, hardware or mechanical constraint such as observations with the same filter, etc.). Blocks of observations in plan schemes are represented by objects (in the sense of object-oriented languages) with attributes corresponding to all the necessary information to perform the observations (absolute target coordinates, user preferences, etc.).

A simplified example of plan scheme is given in figure 1. It should be noted that some blocks that may be observed at different times of the night may be redundantly scheduled (especially if they are very critical) in order to give them more chances (like observation O4 in the figure). In the general case, a block belongs to several sectors. Indeed, along the whole duration of a block of observation it can "jump" from one sector to another.

This approach provides a pre-computed provision for mechanical or visibility problems. During the night, the dynamic execution can browse the plan scheme and switch from one sector to another if necessary. Switches may result from various events that can occur during a night, either under normal conditions (termination of a block of observations and highest priority block in another sector) or because of a perturbation (clouds in one sector or alert).

The plan scheme structure can accommodate Targets of Opportunity (GRB) with little computation. The occurrence of a TOO does not modify the existing execution lines, it just adds a new one with high priority observations concerning the TOO. Hence, if these observations are impossible, other ones (that still are in the plan scheme) can take place with no re-scheduling at all.

4.2. During the Night

The nightly module must react quickly to changes in observing conditions, observation failures, or unforeseen events and it must modify the observation plan scheme if necessary (in the following we shall call such a system *reactive* with respect to its environment). Moreover, the execution of any block of observations in the plan scheme can be interrupted if necessary, either because of an emergency or to accommodate Targets Of

=> periodic observation PO1has priority

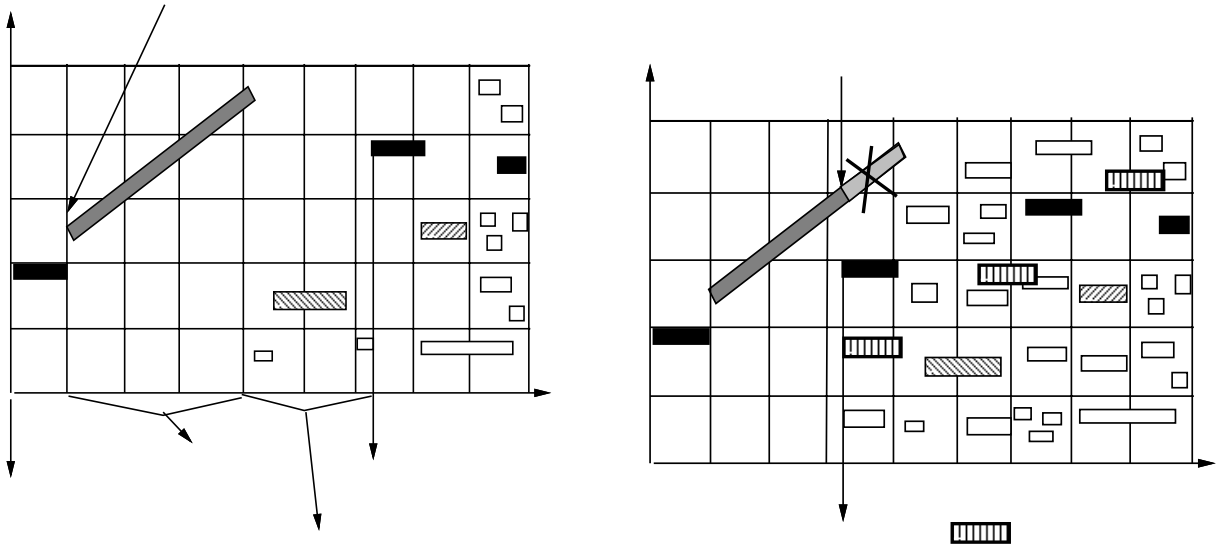
end of tracking & better visibility in sectors 4, 5 => observe in these sectors

correct visibility in sectors 1 2 =>PO1−3

=> highest priority GRB observation preferred

to follow GRB

Opportunity. Figure 2 shows examples of what could happen to the plan scheme of figure 1 during two different nights. The first night execution 2(a) correspond to rather ideal conditions, choices are made in the plan based on priority and visibility conditions. The second night execution 2(b) shows an observation block (tracking TO) aborted because a higher priority TOO occurs.



(a) A night with few clouds.

(b) A night with a GRB.

Figure 2. Examples of executions in reaction to different conditions and events

The nightly module is a *reactive* software, its implementation is event-driven. It does not stick to time-tagged commands, but rather reacts to a various events. In figure 2(a), the events are "start" or "termination" of a (block of) observations (such as PO1-1), "good visibility in sector X", etc., in 2(b), another event occurs: "Gamma Ray Burst".

For these reasons, we have decided to apply reactive models and techniques, as explained in the following section.

4.3. Implementation of Nightly Module

Among reactive techniques, we have chosen to use the SyncCharts[7] graphical modeling language. SyncCharts allow to express a complex reactive behavior by simply drawing automata that (logically) communicate by broadcasting signals. SyncCharts favor extensibility: automata can be refined by means of hierarchical decomposition and concurrency. SyncCharts also support different kinds of preemptions: abortion and suspension of physical processes.

The main features supported by SyncCharts are detailed below.

- *Event-driven* systems. In this model, a system reacts to occurrences of signals (the events) by producing other events. Events may be used internally (by a sub-automata) or come from (resp. be emitted to) external systems, via a connection to sensors (resp. to actuators).

- *Reactivity.* SyncCharts only model the control part of a system, i.e. events reception and sending. The computational parts (algorithmic functions) are out of the model, but they can be connected to it.
- Representation based on deterministic *state-transition model* (automata). A reaction to an event is the transition from the current state to another one, plus the possible execution of some actions. Actions (that produce events) may occur on a transition or inside a state, if the target state of the transition has some inner behavior.
- *Concurrency.* Several automata can execute simultaneously. Signals are used to synchronize automata or to transmit information among automata states (for example in of Fig. 3, the `propose` signal transmits a value from PROPOSER to DECIDER).
- *Hierarchy.* Every state may be refined at any level, either as a more precise sub-automaton or eventually as a link to a piece of executable code.
- Preemption or normal termination allow to leave a state aborting whatever was going on and suspension makes it possible to "freeze" the state internal behavior.

This technique implies to know the signals in advance, because no dynamic generation is possible. In our case, fortunately, the signals, i.e. the *types* of possible events are known (although not the time and number of occurrence of each type) and the corresponding reactions are also known. For instance, in reaction to any occurrence of a GRB in an observable sector, astronomers want to include periodic observations of the region of the sky where the burst has been detected. These observations have a higher priority than other ones and are to be performed during the current night and the following ones. We can hence describe each event/reaction of the nightly module in the form of a transition and a target state –reachable on reception of this event– with the appropriate refined inner behavior.

Tools exist to design SyncCharts, to translate them into computer languages, and even to provide a simulation of their behavior. Thanks to this rich programming environment, programmers can design high-level behavioral descriptions of programs in the form of automata and simulate them. It is also possible to associate pieces of code with a state of an automaton, that implement desired algorithmic actions to perform inside this state. Once satisfied, the programmer can automatically translate his/her SyncCharts into a computer program, that exhibits the *same* behavior as its specification automata.

Our nightly module fits perfectly in this model: it must have a reactive behavior in a changing environment, there are various kinds of events that should trigger different actions, some events (TOO's) are preemptive, etc. Indeed, SyncCharts have already been applied to complex control applications (a machine-tool or a sub-marine mobile robot, for instance). Moreover, the code generated from the automata can be linked with any computer program to implement the necessary algorithmic actions. Hence SynchCharts provide a very modular way of creating a new scheduler: it is easy to add/remove an automata (i.e. a partial behavior as a reaction to a signal) or to change a computation function inside a state. To this end a recompilation and a link are enough.

Figure 3 shows a high-level view of the nightly module in the form of automata. The scheduling part (state SCHEDULER) is refined into 2 concurrent sub states. First, the PROPOSER reacts to every event (except the one it emits) and then triggers an action that scans the current plan scheme to sort the blocks of observations that are feasible at this time in all sectors, i.e. blocks that come after the current time in the plan scheme and still have enough time to execute. The result is a list of blocks (L). Second, the DECIDER when receiving the `propose` signal with L as a value, triggers an action that selects the best observation to perform in L, based on the current situation. It emits the `execute` signal (awaited by EXECUTOR to start) with the next block of observation to perform (BO) as value. The EXECUTOR permanently emits its status (idle or working). A GRB occurrence preempts the SCHEDULER. If the GRB zone is visible, the GAMMA state emits a `kill` event that aborts the execution of the current block of observations. GAMMA internal actions always modifies the current plan scheme, adding new observations related to the GRB.

The nightly module hence exhibits a flexible architecture: if a new behavior has to be added in reaction to a new signal, the automaton of this new behavior should just be "plugged" into the existing ones, either by adding a concurrent automaton or by refining a state.

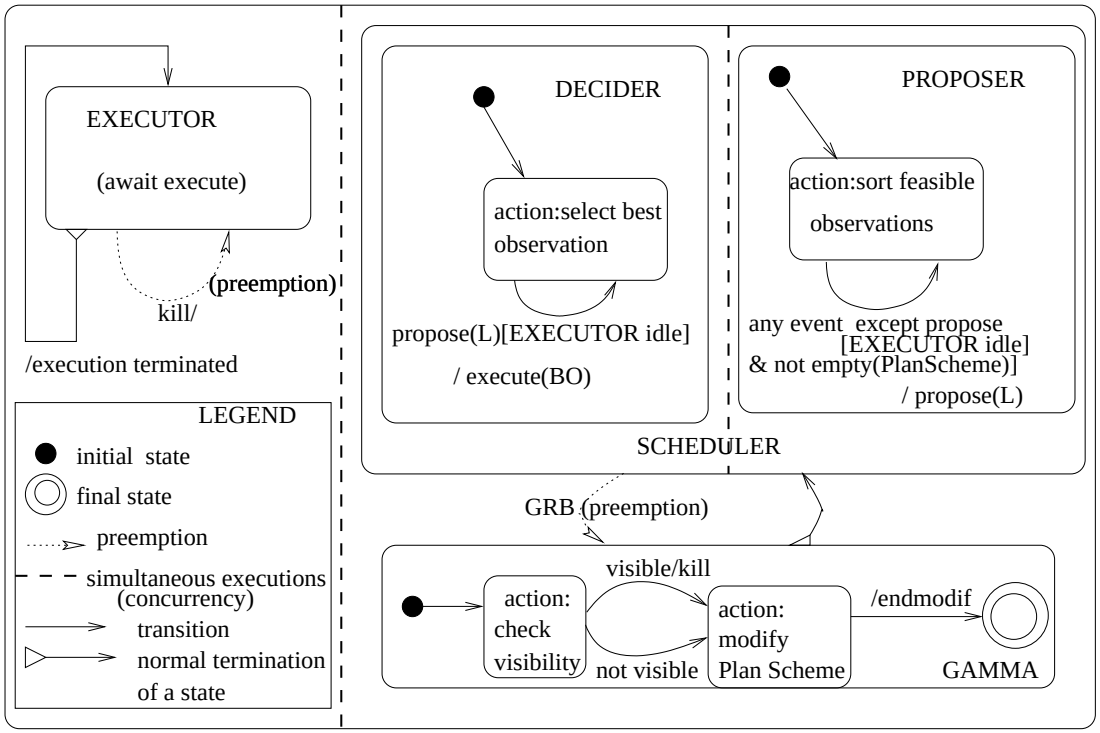


Figure 3. High-level view of the nightly module SyncCharts. Each transition has a *label* representing an elementary execution step of the entity, consisting of a *trigger* (input event(s) and possibly condition inside squared brackets) and the *action* to be executed when the transition is fired. In our case an action often corresponds to the emission of an event. A label is noted `input [condition] / output`.

This module is associated with a visualization interface to graphically display the plan scheme, the current execution line, and the reorganizations due to TOO's or problems. Successful observation results are stored in a database, to be sent to the image processing module, which in turn sends its results to the proposers. Unsuccessful observations, not performed during the night for any reason, are stored in a data base to be planned again later, if possible. This reorganization may lead to modify the priorities of some observations, mainly according to their criticity (time remaining in the observation window or number of previous observations, if the observation is periodic). If they can not be planned again (e.g., if they have reached the end of their observation window) they are canceXSled. We also foresee that the results from the data processing pipeline may directly produce new requests and/or TOO's.

4.4. Comparison with other Work

The importance of automated planning and scheduling for the space enterprise has become increasingly clear,[8,9] especially with the advent of new telescopes.

Several scheduling algorithms and software have been developed for observation and mission planning, like e.g., ATIS-Dispatch algorithm,[10,11] ASTER[12] or Hobby[13] schedulers. Spike[14] is one of the earlier example, it is a general framework for planning and scheduling based on constraint satisfaction techniques. It is used by many astronomical observatories and systems such as HST, ESO VLT,[15,16] Subaru, Chandra, and others.

Most schedulers are intended to be used as expert *assistants* for astronomers or telescope staff. An example is SEA[17] (NASA's Goddard Space Flight Center and the Space Telescope Science Institute) whose aim is to reduce time and effort in proposal preparation for both scientists and telescope operation staff, thanks to visual tools. It thus provides scientists with a software interface to guide them in the preparation of their observing proposal. Another example in the Gemini project,[18] is the Observing Tool (OT) which offers astronomers an

help for preparing observations. Astronomers use the observing tool while planning observations and configure the observatory systems thanks to the OT. They can also examine the progress of a science program at any time. The Associate Principal Astronomer (APA)[19] for Telescope Operations is another support tool for multiple astronomers accessing a remote automatic telescope over the Internet. It is primarily based on ATIS-Dispatch scheduler.[10] These assistants avoid tedious manual planning, but they still need a person (e.g., a Principal astronomer) who is in charge of collecting the requests, sorting them and following up the execution. On the contrary, we aim at full automation of the whole process.

Generally, the scheduling systems rely on re-scheduling to accommodate weather evolution, TOO's, etc. For instance, Spike scheduling engine has been adapted for Subaru[20] by adding a real-time scheduling algorithm to re-schedule observations. But such dynamic re-scheduling may be costly and waste too much time, to the detriment of scientific observations. That is why a usual complementary approach is to build schedules as robust as possible, that tend not to break because they are prepared for likely errors. It is the case for Just-In-Case Scheduling.[21] This algorithm perform off-line re-scheduling to generate a contingent schedule to cover possible breaks. It focuses on uncertainty regarding action duration that could cause execution breaks. In the same line, the notion of contingent plans[22] has appeared more recently. But such approaches lead to a lot of processing that may be useless if breaks do not occur when planned.

We follow the same idea to avoid dynamic re-scheduling by predictive error-management, but with a different approach. At any time when an observation is possible we ensure that the best choice is selected (with respect to astronomers' predefined choice criteria), without spending time for re-scheduling. First, we do not compute a real plan with pre-scheduled branching points in case of problem, but rather a plan scheme where free navigation is possible. Second, we rely on dynamic decisions during the night to execute the best possible observations. This approach does not introduce useless processing during the night, only the necessary computation is done (for instance, where to plan new observations for a given GRB). In our architecture, the only pre-programmed part is one "handler" per problem *type* (i.e. per signal). The reactive program calls these handlers only when necessary, if and only if a problem of the corresponding type actually occurs. This allows us to accommodate not only uncertainty in durations, but also TOO's, visibility, hardware problems, etc.

5. CONCLUSION AND FUTURE WORK

We propose completely automatic tools different from those that are used by astronomers or telescope staff to construct observing plans to be executed by the telescope. The proposed approach is not committed to ARAGO and is general enough to be applied to any other type of telescope, provided that reactivity is important. In our design neither the duration, the type of constraint, the type of observed object, nor the instrument at focus play a role. For instance, long spectroscopical frames or observations of orbital debris may be scheduled as well.

Our primary concerns are modularity (for easy reconfigurations) and reactivity to a wide range of events. This has been achieved first by combining independent "classical" modules (in C++). Each module may be customized (e.g., parameterized by different strategies to combine observing constraints for ordering). The original use of SyncCharts seems well adapted to increase reactivity, but also to facilitate reconfiguration, because it provides a high-level programming environment where changes are easier to implement than by (error-prone) direct code modification. Moreover, it also provides simulation tools to run different scenarios. This technique could also be used for the control part of the telescope as it has been done at Nice Observatory.[23] In the short term, we will propose a prototype system, test it against real request flows, and compare its results with the previous version of the MAJORDOME.

In the future, we plan to investigate two main research directions. First, we would like to link planning and scheduling tools with the image processing system. This will provide us with a completely integrated system with feedback from previous observations that may influence next night scheduling. Image processing is done in parallel with observations, but since it may take some time, the results are usually not available in real-time. On the one hand, a poor image processing result may lead to re-schedule a past observation, on the other hand, discovering an interesting object in an image may generate new observation of the same area for future night(s). Such an automatic feedback would be of great help for the telescope users. Second, our intent is also to use artificial intelligence (AI) techniques, that have been already experimented in observing tools,[24] based

on our long experience in knowledge-based systems.[25,26] Introducing these techniques can increase flexibility and adaptability in our system. But before doing that, a control model as proposed in this paper was necessary. It will help us identify where AI could be usefully introduced. For instance knowledge-based techniques could improve and simplify the daily construction of plan scheme by means of heuristics and rules that represent knowledge about observations, preferences, choice strategies, etc. Such knowledge may help in pruning the global search tree. Furthermore, astronomers could themselves tune the daily tool, because modifying this knowledge does not require programming expertise.

ACKNOWLEDGMENTS

D. Gaffé (I3S/Univ. Nice) for his help in SynchCharts use.

REFERENCES

1. A. M. Chavan, G. Giannone, D. R. Silva, A. P. Krueger, and G. E. Miller, "Observatory Operations to Optimize Scientific Return," in *SPIE Proceedings Series*, P. J. Quinn, ed., **3349**, 1998.
2. J. Bresina, R. Morris, and W. Edgington, "Optimizing observation scheduling objectives," in *First NASA Workshop on Planning and Scheduling for Space Exloration and Science*, (Oxnard, CA.), 1997.
3. M. Boër, "Agile telescopes to monitor optical transients and sky variability: From TAROT to ARAGO," *Astronomichal Nachrichten* **322**, pp. 343–346, 2001.
4. M. Bringer, M. Boer, C. Peignot, G. Fontan, and C. Merce, "Flexible automatic scheduling for autonomous telescopes: The majordome," *Experimental Astronomy* **12**, pp. 33–48, 2001.
5. M. Bringer, M. Boer, C. Peignot, G. Fontan, and C. Merce, "Flexible Automatic Scheduling For Autonomous Telescopes: The MAJORDOME," in *Astronomical Data Analysis Software and Systems IX*, D. C. N. Manset, C. Veillet, ed., *ASP Conference Series* **216**, Kluwer Academic Publishers, 2000.
6. M. Boër, J. L. Atteia, M. Bringer, B. Gendre, A. Klotz, R. Malina, J. A. de Freitas Pacheco, and H. Pedersen, "Limits on the early afterglow phase of gamma-ray burst sources from TAROT-1," *Astronomy and Astrophysics* **76**(378), 2001.
7. C. André, "Representation and Analysis of Reactive Behaviors: A Synchronous Approach," in *CESA'96 Proceedings*, IEEE-SMC, (Lille, France), 1996.
8. M. D. Johnston, "Scheduling Tools for Astronomical Observations," in *ASP Conf. Ser. 87: New Observing Modes for the Next Century*, pp. 62+, 1996.
9. A. M. Chavan, M. D. Johnston, and M. A. Albrecht, "Building Scheduling Tools for Ground Based Telescopes," in *ASP Conf. Ser. 87: New Observing Modes for the Next Century*, pp. 58+, 1996.
10. G. W. Henry, "ATIS Dispatch Scheduling of Robotic Telescopes," in *New Observing Modes for the Next Century*, T. A. Boroson, J. K. Davies, and E. I. Robson, eds., *ASP Conference Series* **87**, 1996.
11. G. W. Henry, "Techniques for Automated High-Precision Photometry of Sun-like Stars," *PASP* **111**, pp. 845–860, July 1999.
12. H. Muraoka, R. Cohen, T. Ohno, and N. Doi, "ASTER Observation Scheduling Algorithm," in *International Symposium Space Mission Operations and Ground Data Systems*, 1998.
13. N. Gaffney and M. Cornell, "Planning and Scheduling Software for the Hobby Eberly Telescope," in *Astronomical Data Analysis Software and Systems VI*, G. Hunt and H. E. Payne, eds., *ASP Conference Series* **125**, 1997.
14. M. Johnston and G. Miller, "SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations," in *Intelligent Scheduling*, M. Fox and M. Zweben, eds., Morgan-Kaufmann, 1994.
15. A. M. Chavan, G. Giannone, D. Silva, and D. Krueger, T.and Miller, "Nightly Scheduling of ESO's Very Large Telescope," in *Astronomical Data Analysis Software and Systems VII*, *ASP Conference Series* **145**, 1998.
16. G. Giannone, A. M. Chavan, D. R. Silva, A. P. Krueger, and G. E. Miller, "Long and Short Term Scheduling Tools in ESO," in *ASP Conf. Ser. 216: Astronomical Data Analysis Software and Systems IX*, **9**, pp. 111+, 2000.

17. K. Wolf, C. Li, J. Jones, D. Matusow, S. Grosvenor, and A. Koratkar, "The Scientist's Expert Assistant Simulation Facility," in *Astronomical Data Analysis Software and Systems (ADASS)*, (Boston, Massachusetts), 2000.
18. S. Wampler, K. Gillies, P. Puxley, and S. Walker, "Science Planning for the Gemini 8m Telescopes," in *SPIE*, **3112**, 1998.
19. M. Drummond, J. Bresina, W. Edgington, K. Swanson, G. Henry, and E. Drascher, "Flexible Scheduling of Automatic Telescopes over the Internet," in *Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy*, G. Henry and J. Eaton, eds., *ASP Conference Series* **79**, 1995.
20. T. Sasaki, G. Kosugi, J. A. Kawai, T. Kusumoto, N. Koura, R. Hawkins, L. Kramer, A. P. Krueger, and G. E. Miller, "Observation scheduling scheme for the Subaru telescope," in *Proc. SPIE Vol. 4009, p. 350-354, Advanced Telescope and Instrumentation Control Software, Hilton Lewis; Ed.*, **4009**, pp. 350–354, June 2000.
21. K. Swanson, J. Bresina, and M. Drummond, "Just-In-Case Scheduling for Automatic Telescopes," in *Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry*, W. Buntine and D. H. Fisher, eds., pp. 10–19, 1994.
22. J. Bresina and R. Washington, "Robustness via Run-time Adaptation of Contingent Plans," in *AAAI-2001 Spring Syposium: Robust Autonomy*, (Stanford, CA), 2001.
23. G. Assman and D. Gaffé, "L'équatorial Coudé : Un système réactif temps-réel conçu par l'approche synchrone," in *Bulletin n. 32 de l'ADION*, O. de la Cote d'Azur, ed., 1998.
24. K. Wolf, C. Burkhardt, M. Fishman, S. Grosvenor, J. Jones, A. Koratkar, and L. Ruley, "Expert System Technology in Observing Tools," in *SPIE Symposium on Astronomical Telescopes and Instrumentation*, 2000.
25. C. Shekhar, S. Moisan, R. Vincent, P. Burlina, and R. Chellappa, "Knowledge-based control of vision systems," *Image and Vision Computing* **17**, pp. 667–683, May 1999.
26. S. Moisan, A. Ressouche, and J.-P. Rigault, "BLOCKS, a Component Framework with Checking Facilities for Knowledge-Based Systems," *Informatica, Special Issue on Component Based Software Development* **25**, pp. 501–507, November 2001.