IST-2001 34140

Schedulability Analysis Tool

Deliverable D-SI.4v2

Responsible: Universidad de Cantabria

By: Michael González Harbour

1 Introduction

The FIRST Scheduling Framework is based on service contracts that provide a guaranteed minimum bandwidth to the application components in the form of a minimum execution time, requested with a given period, and with a deadline specified in the contract. This minimum guarantee is used by the application component to predict whether or not it will meet its hard real-time requirements.

In the second phase of the project there is a one to one mapping of application threads to service contracts, and this make it possible to do the analysis in a trivial way, using the standard schedulability analysis tools that exist for the underlying scheduling mechanism, for instance fixed priorities or EDF.

However, in the third phase of the project, two levels of hierarchical scheduling are planned: Several threads may be using the same service contract, and will be scheduled with a secondary scheduler built into the FSF implementation. For the moment, the secondary scheduler will be a fixed priority or EDF scheduler, although other alternatives may be explored in the future. The analysis for this kind of hierarchical scheduling was developed in Phase 1 of the project. In Phase 2, we have implemented into a schedulability analysis tool the analysis for hierarchical scheduling with fixed priorities as the underlying mechanism. This implementation is presented in this document.

In addition to the tool implementation, most of the restrictions on the schedulability analysis developed in Phase 1 of the project were eliminated. These restrictions were independent periodic tasks with deadlines less than or equal to periods, executing on single processors. We now allow task synchronization (through the stack resource policy, SRP [5]), and arbitrary deadlines. Analysis for distributed systems with mixed EDF and Fixed priorities is planned for Phase 3 of the project.

The mixture of table driven and fixed priority scheduling does not need new schedulability analysis techniques, because the timing requirements are already imposed in the table by construction, and the methodology used to transform the table into a set of tasks with given priorities and offsets guarantees that the timing requirements will be met.

2 The MAST modeling and analysis suite for real-time systems

MAST [8] is the acronym for Modeling and Analysis Suite for Real-Time Applications, and is an open source set of tools that enables engineers developing real-time applications to check the timing behaviour of their application, including schedulability analysis for checking hard timing requirements. It has been developed by the University of Cantabria, Spain[1].

A model for describing real-time applications should represent not only the characteristics of the architecture of the distributed system, but also the hard real-time requirements that are imposed. Most of the existing analysis techniques for the scheduling of distributed hard real-time systems are based on a model that we call *linear*, which is representative of a large number of systems. In the linear model each task is activated by the arrival of a single event or message, and each message is sent by a single task. However, this linear model does not allow complex interactions among the responses to different event sequences, except for the shared resource synchronization, and so, the analysis is not applicable to systems in which these interactions exist.

The MAST model uses a very rich representation of the real time system. It allows single-processor systems as well as multi-processor and distributed systems. It is an event-driven model in which complex dependence patterns among the different tasks can be established [1]. For example, tasks may be activated with the arrival of several events, or may generate several events at their output. This makes it ideal for analysing real-time systems that have been designed using object-oriented methodologies, and event-driven architectures. Indeed tools have been developed for automatically obtaining a MAST model description of a real-time application from a standard UML description to which a real-time view of the system has been added [7].

1. Available from: `http://mast.unican.es/`

The MAST suite includes schedulability analysis tools that use the latest offset-based techniques [2][3] to enhance the results of the analysis. For distributed systems, these techniques are much less pessimistic than previous schedulability analysis techniques [4], which are also included in the toolset for completeness. The toolset also includes tools for assigning optimized priorities, and for the simulation of the timing behaviour of the system.

The MAST toolset is open source and is fully extensible. That means that other research teams may provide enhancements. The first versions were intended for fixed priority systems, but with the extensions performed in this workpackage of the project, dynamically scheduled systems and hierarchical scheduling are supported.

The model is included in a toolset (Figure 1 and Figure 2), with the following elements:

- The model and the results are specified through an ASCII description that serves as the input and output of the analysis tools. Two ASCII formats have been defined: a concise special-purpose format and an XML-based format.
- Graphical editors and other tools generate the system using one of these ASCII descriptions. They can then invoke the analysis tools.
- A parser converts any of the ASCII descriptions of the system into an Ada data structure that is used by the tools. A module is offered to convert the Ada data structure back into the chosen ASCII description.
- The XML format provides the designer with capabilities to use free standard XML tools to validate, parse, analyse, and display the model files.
- A results viewer is available to view the analysis results in a convenient way.

The MAST environment will integrate the following tools described in Figure 1:

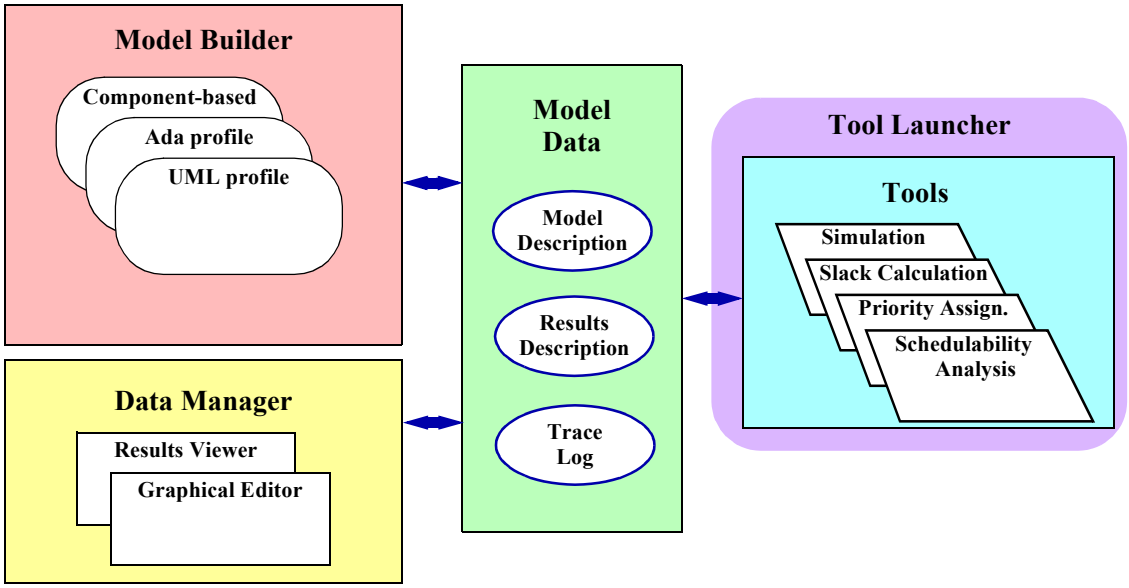

Figure 1. MAST toolset environment

- The schedulability analysis tools perform different kinds of worst-case analysis to determine the schedulability of the system. Blocking times relative to the use of shared resources are calculated automatically.

- The priority assignment tools are able to make an automatic assignment of priorities and priority ceilings, using optimum priority assignments when available, and heuristics or optimization techniques when the optimum assignment is not available.
- The slack calculation tools calculate the system, processing resource or transaction slacks by repeating the analysis in a binary search algorithm in which execution times are successively increased or decreased.
- The simulation tools are able to simulate the behaviour of the system to check soft timing requirements and generate temporal traces of the simulated execution.

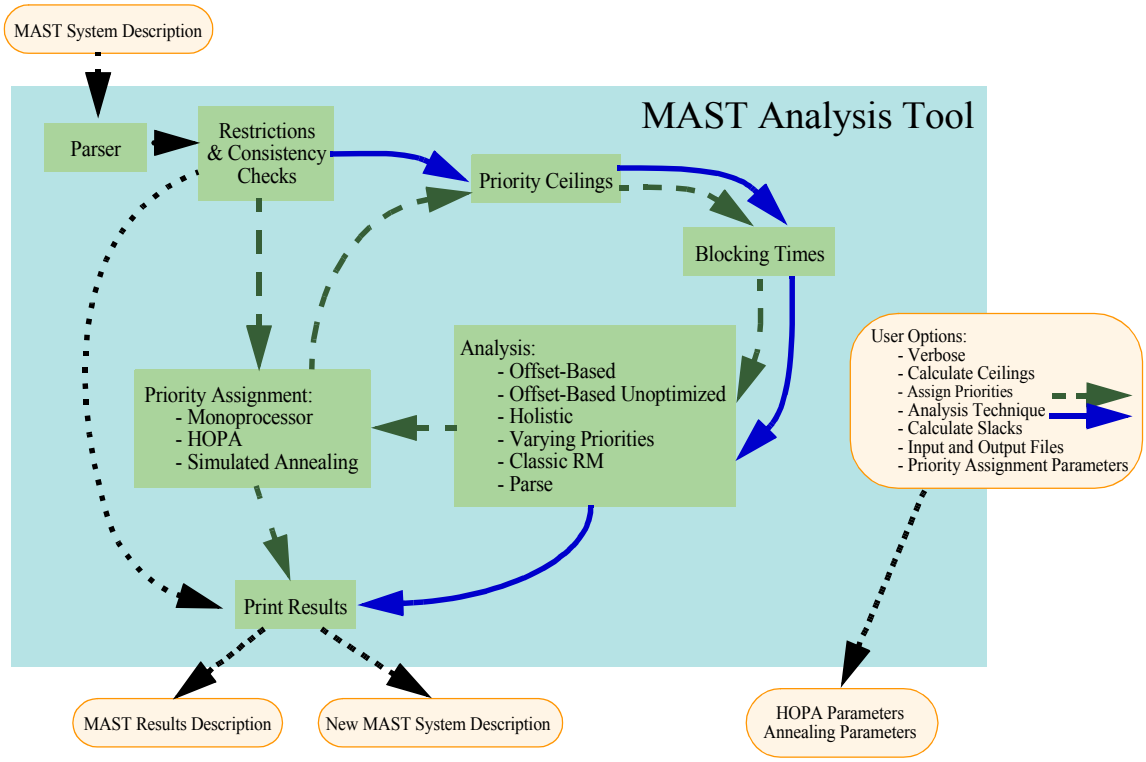

Figure 2. MAST Analysis tools

Using a standard CASE UML tool, it is possible to describe the real-time behaviour of the system by means of a set of appropriate UML modeling primitives that are defined in different profiles in accordance with the technology used to design the real-time system (object oriented, Ada language, component based, etc.). Then, an automatic tool is used to compile the UML real-time view and to build the MAST real-time model description. No special framework is needed with this approach, but the designer must incorporate the real-time view into the UML description. Refer to the UML-MAST project page[1] for more information of this issue.

Figure 2 represents the MAST toolset. The capabilities of the currently implemented tools are represented in the following tables. The tools with dark shading are still under development.

1.http://mast.unican.es/umlmast/

Table 1: Fixed-priority schedulability analysis tools

Technique	Single-Processor	Multi-Processor	Simple Transact.	Linear Transact.	Multiple Event T.
Classic Rate Monotonic	☑		☑		
Varying Priorities	☑		☑	☑	
Holistic	☑	☑	☑	☑	
Offset Based Unoptimized	☑	☑	☑	☑	
Offset Based	☑	☑	☑	☑	
Multiple Event	☑	☑	☑	☑	☑

Table 2: EDF schedulability analysis tools

Technique	Single-Processor	Multi-Processor	Simple Transact.	Linear Transact.	Multiple Event T.
Single Processor	☑		☑		
EDF_Within_Priorities	☑		☑		
Holistic	☑	☑	☑	☑	
Offset Based	☑	☑	☑	☑	

3 The MAST model for real-time systems

A real-time situation is modelled as a set of concurrent transactions that compete for the resources offered by the platform. Each transaction is activated from one or more external events, and represents a set of activities that are executed in the system. Activities generate events that are internal to the transaction, and that may in turn activate other activities. Special event handling structures exist in the model to handle events in special ways. Internal events may have timing requirements associated with them.

Figure 3 shows an example of a system with one of its transactions highlighted. Transactions are represented through graphs showing the event flow. This particular transaction is activated by only one external event. After two activities have been executed, a multicast event handling object is used to generate two events that activate the last two activities in parallel.

We call the “boxes” that are included in the transaction *Event Handlers*. As we have mentioned, there are event handlers that just manipulate events, like the *Multicast* event handler in Figure 3. Another very important event handler is an *Activity*, which represents the execution of an operation, i.e., a procedure or function in a processor, or a message transmission in a network.

The elements that define an activity are described in Figure 4. We can see that each activity is activated by one *input event*, and generates an *output event* when completed. If intermediate events need to be generated, the activity would be partitioned into the appropriate parts. Each activity executes an *Operation*, which represents a piece of code (to be executed on a processor), or a message (to be sent through a network). An operation may have a list of *Shared Resources* that it needs to use in a mutually exclusive way.

The activity is executed by a *Scheduling Server*, which represents a schedulable entity in the *Scheduler* to which it is assigned. This scheduler belongs to a *Processor* or a *Network*, although we will see that when hierarchical scheduling is modelled the situation is somehow more complex. For example, the model for a scheduling server in a processor is a task or thread. A thread may be responsible of executing

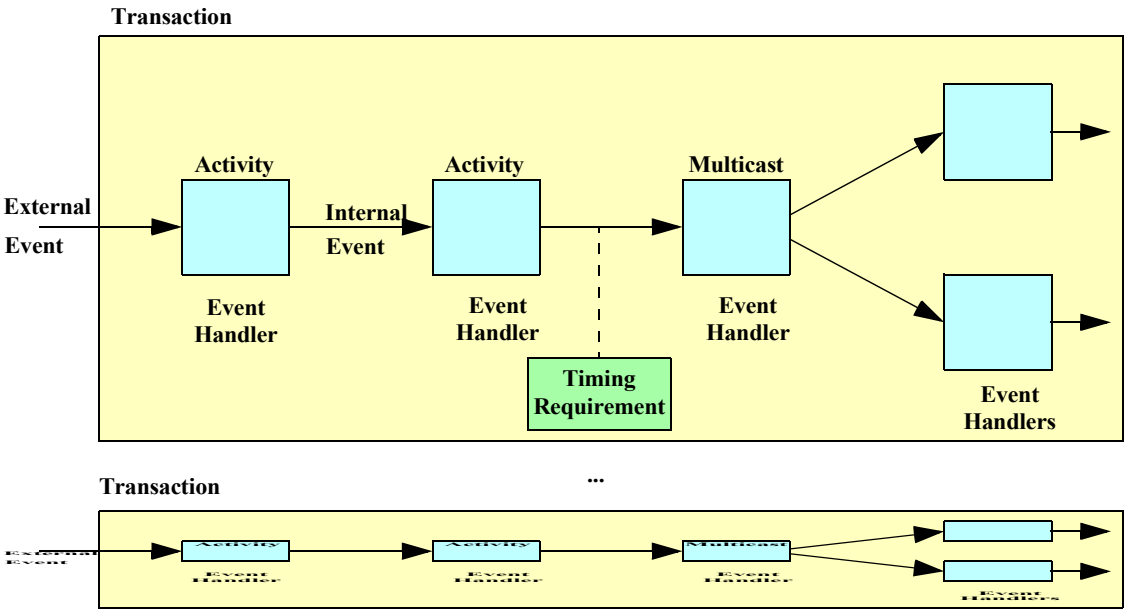

Figure 3. Real-Time System composed of transactions

several activities (procedures). The scheduling server is assigned a *Scheduling Parameters* object that contains the information on the scheduling policy and parameters used.

4 Hierarchical scheduling in MAST

4.1 Schedulers

A scheduler is a MAST object used to represent the operating system objects that implement the appropriate scheduling strategies to manage the amount of processing capacity that has been assigned to them.

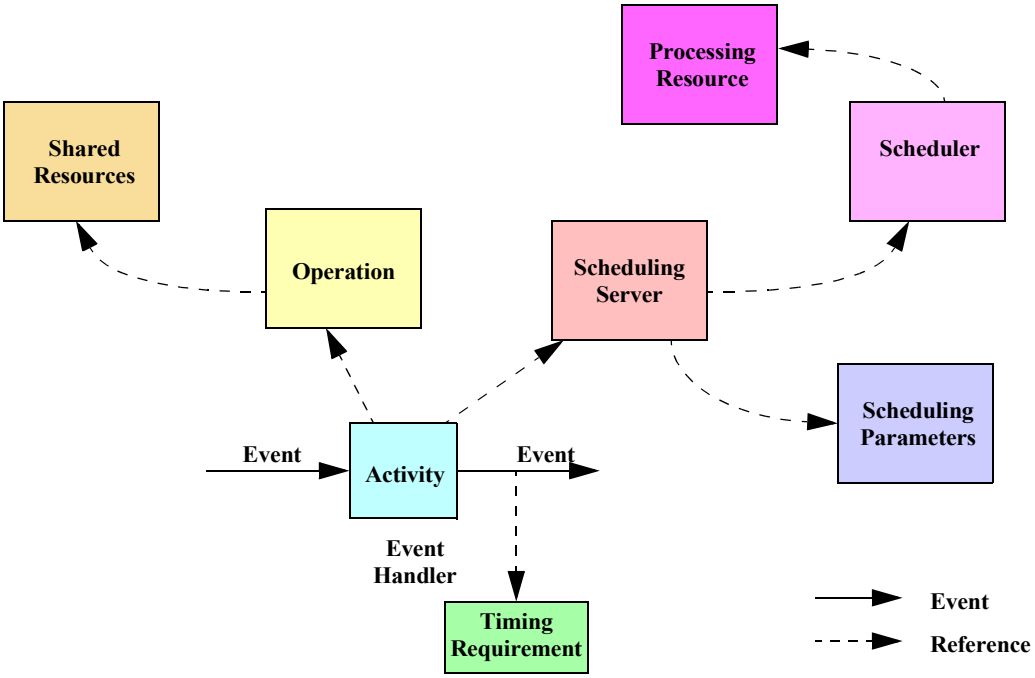

Figure 4. Elements that define an activity

Schedulers can have a hierarchical structure, like the one shown in Figure 5. A *Primary Scheduler* operates by offering the whole processing capacity of its associated base processor to its associated scheduling servers. A *Secondary Scheduler* is allowed to deliver to its scheduling servers just the processing capacity that it receives from its associated scheduling server, scheduled in turn by some other scheduler.

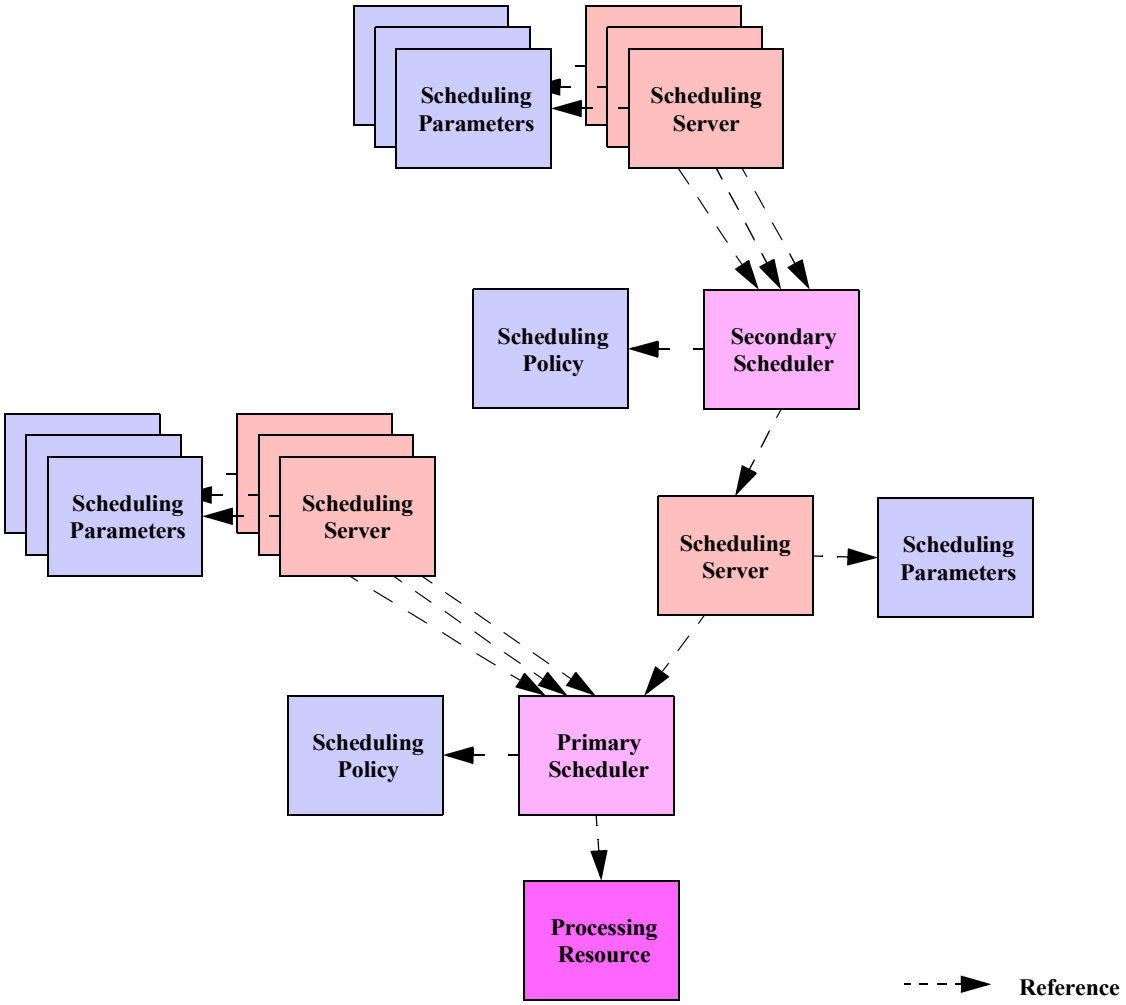

Figure 5. Hierarchical scheduler structure

Therefore, we have two kinds of scheduler:

- *Primary Scheduler*. It represents the base system scheduler for its associated processing resource.
- *Secondary Scheduler*. It represents a scheduler that is able to handle only a certain fraction of processing capacity, which in turn is served by a particular scheduling server through another scheduler associated to it.

4.2 Scheduling policies

A scheduling policy represents the basic strategy that is implemented in a scheduler to deliver the assigned processing capacity to its associated scheduling servers. Each of these servers has a *Scheduling_Parameters* object that describes the parameters used by the scheduler.

Classes of Scheduling Policies:

- *Fixed Priority*. It represents a fixed-priority policy.
- *EDF*. It represents an Earliest Deadline First policy.

- *Fixed Priority Packet Based*. It represents a fixed priority policy used in a packet oriented communication network. Network packets are assumed to be non preemptible.

4.3 Scheduling parameters

These parameters are attached to a scheduling server and are used by its scheduler to make its scheduling decisions when the server is competing with other servers. Some scheduling policies allow several compatible scheduling behaviours to coexist in the system. For example, the fixed priority scheduling policy allows both preemptive and non preemptive activities. These different scheduling behaviours are determined by the scheduling parameters type and are also called the per-server policy parameters.

There are several classes of scheduling parameters. There are some restrictions on the compatibility between scheduling parameters and the scheduling policy of the associated scheduler. The scheduling parameters described in Section 4.3.1 are only applicable to schedulers with the fixed priority policy. Those described in Section 4.3.2 are only applicable to schedulers with the EDF policy.

4.3.1 Fixed priority scheduling parameters

The classes defined are:

- *Non Preemptible Fixed Priority Policy.* Activities scheduled with these parameters are non preemptible.
- *Fixed Priority Policy*. Represents a regular preemptive fixed priority parameters object.
- *Interrupt Fixed Priority Policy*. Represents an interrupt service routine. No additional attributes. The "*Preassigned*" field cannot be set to "*No*", because interrupt priorities are always preassigned.
- *Polling Policy*. Represents the scheduling mechanism by which there is a periodic task that polls for the arrival of its input event. Thus, execution of the event may be delayed until the next period.
- *Sporadic Server Policy*. Represents a task scheduled under the sporadic server scheduling algorithm.

4.3.2 EDF Scheduling Parameters

All the EDF scheduling parameters have the following set of common attributes:

- *Deadline*. Relative deadline of the associated scheduling server. Its default value is a Large Time.
- *Preassigned*. If this parameter is set to the value "*No*", the deadline may be assigned by one of the deadline assignment tools. Otherwise, it is fixed and cannot be changed by those tools. Its default value is "*No*" if no deadline field appears in the MAST description, and "*Yes*" if any of them is present.

There is only one class of EDF scheduling parameters defined at the moment:

- *Earliest Deadline First Policy*. Represents a task scheduled according to the "earliest deadline first" policy.

4.4 Synchronization Parameters

These parameters are attached to a scheduling server to specify the parameters used by that server when performing a mutually exclusive access to shared resources.

The synchronization parameters should be specified whenever the scheduling policy is such that the given *Scheduling Parameters* do not have enough information for the synchronization protocols used. For example, no synchronization parameters are needed for the priority inheritance or immediate priority ceiling protocols, because the only information they require from the server is its priority.

The only class defined for the synchronization parameters is named *Stack Resource Protocol Parameters* because it is associated with that synchronization protocol.

4.5 Scheduling Servers

They represent schedulable entities in a scheduler. There is only one class defined, named *Regular*.

5 Schedulability Analysis

Scheduling analysis describes the process of testing a set of tasks to determine whether they will meet their deadlines if a particular scheduling strategy is used at runtime. *Response time analysis* is a family of schedulability analysis techniques that is based on calculating the worst-case response time of different tasks or activities in the system, and comparing them to the respective deadlines.

In the approach with fixed priorities as the underlying scheduling mechanism, the system is scheduled with hierarchical schedulers, and the root of the hierarchy is based on fixed priorities. EDF schedulers can schedule tasks at a given priority level. We have implemented the EDF-Within-Priorities analysis developed in Phase 1 of the project and published at the last RTSS [6]. It is now one of the response time analysis tools supported by MAST.

The EDF-Within-Priorities analysis can also be used for regular EDF systems, because we can assume that there is only one EDF scheduler executing at the lowest priority level. We can also take into account the effects of interrupts, because they can be modelled as high priority fixed priority tasks executing before any EDF tasks. Therefore, the tool has also been used to implement the single processor EDF response time analysis in MAST.

The stack resource protocol (SRP) is used to synchronize EDF tasks, and its blocking time calculation is now included in the MAST tool. If it is necessary to synchronize EDF and fixed priority tasks, or EDF tasks belonging to different schedulers, then the immediate priority ceiling should be used with the appropriate priority ceilings. Correct interoperability between these synchronization policies is handled by the tool.

We have also added deadline and preemption level automatic assignment capabilities to the tool. The scheduling deadlines are just copied from the associated timing requirements, avoiding the need for the user to specify both. Preemption levels are fully assigned by the tool. If a manual assignment is preferred, the tool checks that the levels are in agreement with the requirements for ordering the tasks according to their deadlines and release jitters.

6 Example

In this section we will show an example of a system that uses hierarchical scheduling mixing FP and EDF tasks, to show how the analysis presented in previous sections can be applied.

Consider a simplified robot controller that has been designed with two concurrent tasks scheduled under fixed priorities. The higher priority task takes care of controlling the servomotors of the robot joints. The lower priority task displays the status of the robot on the screen. The timing requirements of these two tasks are shown below. The tasks have no release jitter:

Task	T_i (ms)	C_i (ms)	d_i (ms)
T1: Servo_Control	5	1	5
T3: Display	1000	100	1000

After the robot controller has been built and tested, the system is extended by adding a new application that processes video images related to the robot environment. This application was developed using an

EDF scheduler and contains three concurrent tasks with the requirements shown below. As before, there is no release jitter:

Task	T_i (ms)	C_i (ms)	d_i (ms)
T21	10	2	10
T22	20	5	45
T23	50	10	50

To ease the integration, the system will be scheduled with a hierarchical scheduler, with a mixture of FP and EDF tasks. The three image processing tasks will be scheduled under their original EDF scheduler at the same FP level. Because the deadlines are between those of tasks T1 and T3, the priority level is chosen to be between the priorities of those two tasks. Therefore, the priority structure of the system is the one shown in Figure 6.

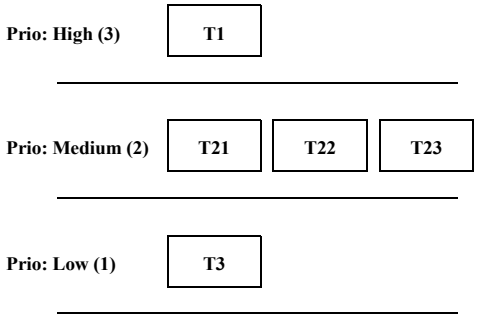

Figure 6. Hierarchical scheduling for the example

We will use the graphical editor tool to develop the real-time model of the system, on which we can apply the schedulability analysis tool. Figure 7 shows the "processing resources" view of the model with a primary FP scheduler and a secondary FP scheduler, and where the attributes of the EDF scheduler have been displayed in a separate window.

Figure 8 shows the scheduling servers, representing the tasks of the system, each linked with its scheduler.

Figure 9 shows a diagram of the operations. In this simple case there is no synchronization, and thus all the operations are simple. The attributes of one of the operations, task 23, are shown; this is where the worst-case execution time of the operation is included.

Figure 10 shows the different transactions, with one of them fully displayed. We can see the input event, where the period or other attributes are defined, and the output event that holds the timing requirements. The other transactions all have a similar structure

Figure 11shows the screen used to invoke the MAST analysis tool, showing the filename, the tool used (EDF_Within_Priorities) and the different options: calculate deadlines and preemption levels, and calculate slacks. The figure also shows the initial screen of the results viewer, showing how the system meets all of its timing requirements, with a slack (in green) of 4.69%. This means that if we increase the execution times of all the operations in the system by that percentage, the system would still be schedulable.

Finally, Figure 12 shows the slacks of each transaction, and the timing results, which are the worst-case response times of each of the tasks in the system. We can see that all of them are schedulable, as marked in green.

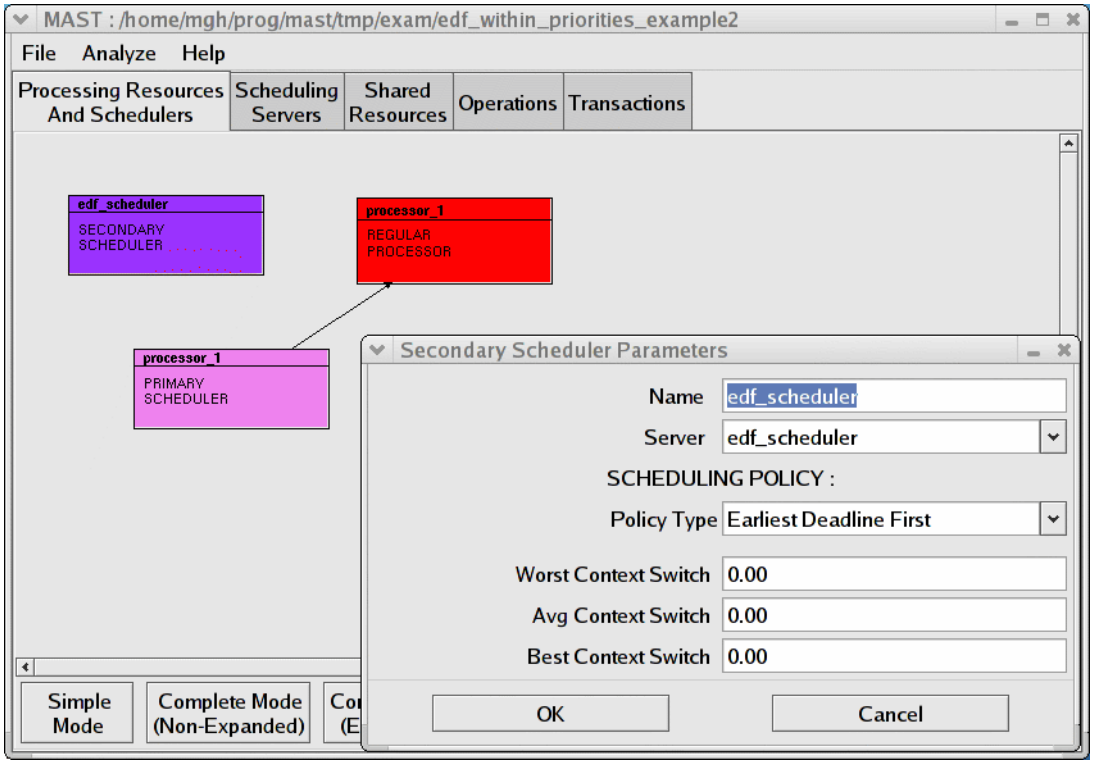

Figure 7. Processing resources and schedulers

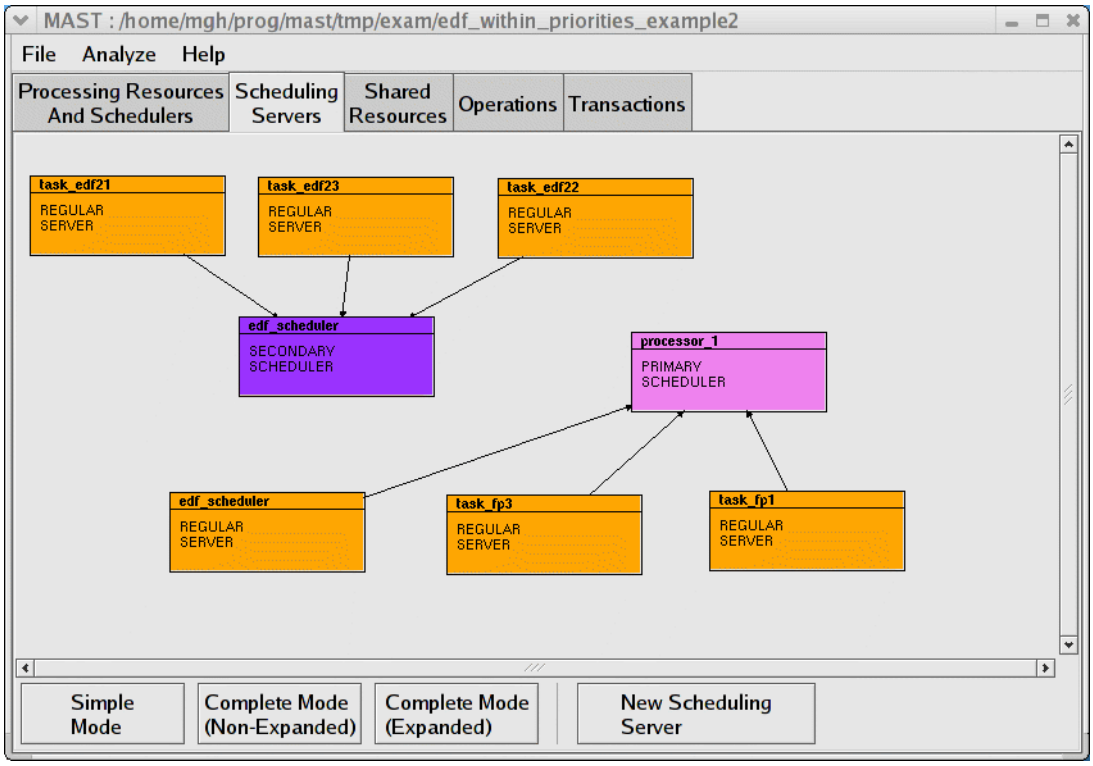

Figure 8. Scheduling servers

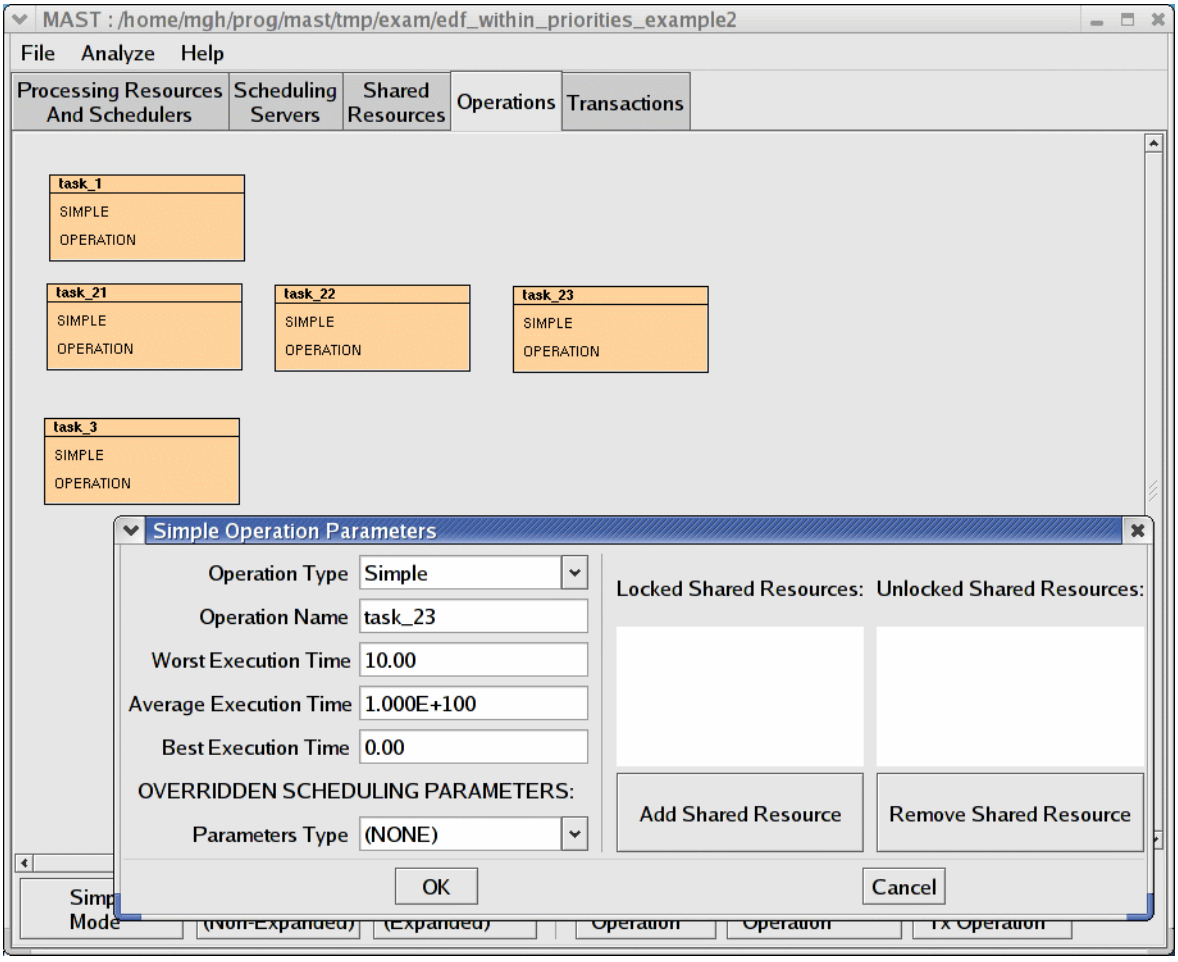

Figure 9. Operations

7 Conclusion

We have extended he original MAST analysis tools with EDF scheduling, the SRP synchronization protocol, and hierarchical scheduling using fixed priorities underneath. These policies are now fully supported for single processor systems, and work is being carried out to provide a full support for distributed systems.

The new tool is available from the tool Web page (`http://mast.unican.es`) as version 1.3.6.

8 References

[1] J.J. Gutiérrez García, J.C. Palencia Gutiérrez, and M. González Harbour, “Schedulability Analysis of Distributed Hard Real-Time Systems with Multiple-Event Synchronization”. Euromicro Conference on Real-Time Systems, Stockholm, Sweden, 2000.

[2] J.C. Palencia, and M. González Harbour, “Schedulability Analysis for Tasks with Static and Dynamic Offsets”. Proc. of the 19th IEEE Real-Time Systems Symposium, 1998.

[3] J.C. Palencia, and M. González Harbour, “Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems”. Proceedings of the 20th IEEE Real-Time Systems Symposium, 1999.

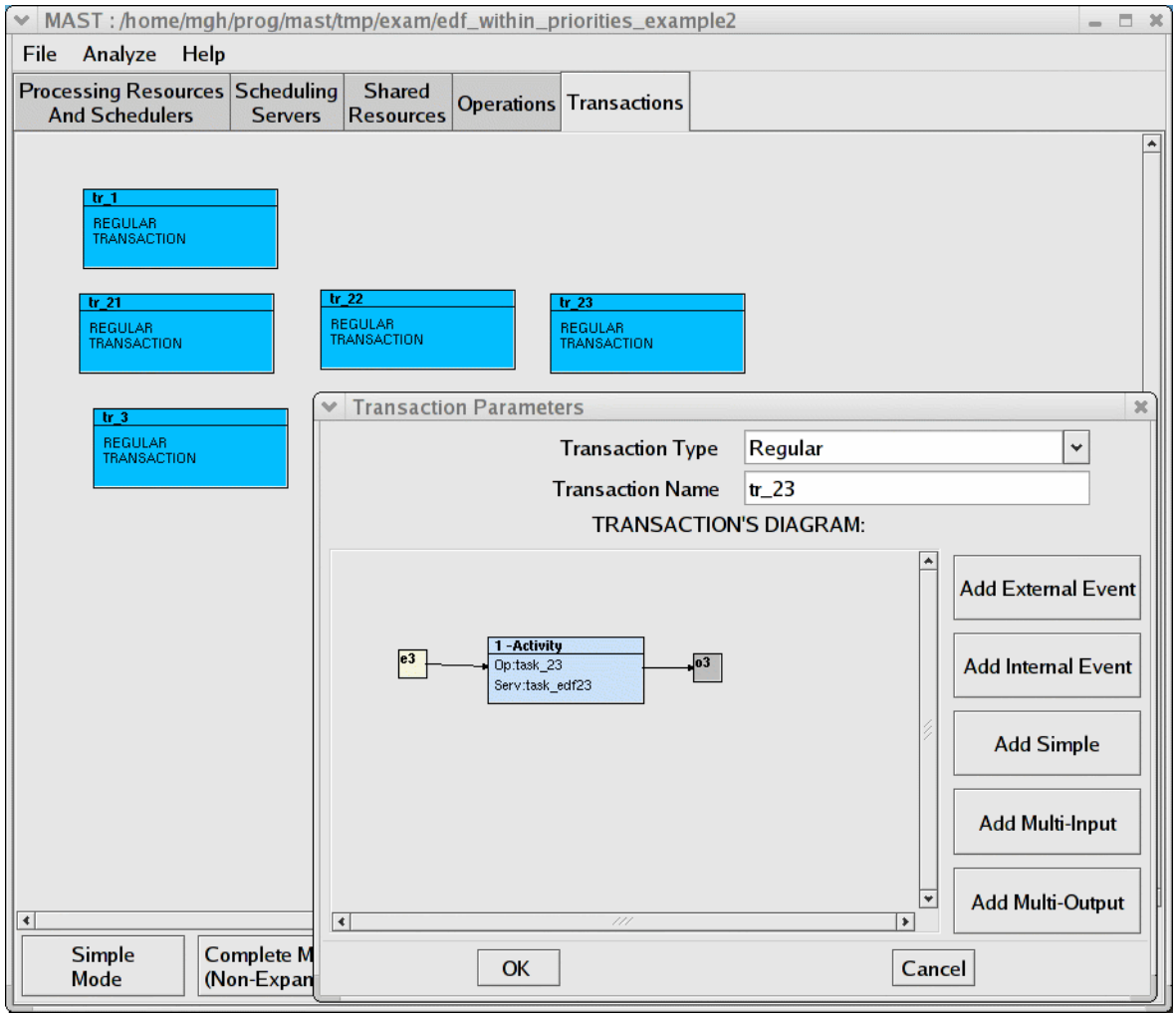

Figure 10. Transactions

[4] K. Tindell, and J. Clark, “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”. Microprocessing & Microprogramming, Vol. 50, Nos.2-3, pp. 117-134, 1994.

[5] Baker T.P., “Stack-Based Scheduling of Realtime Processes”, Journal of Real-Time Systems, Volume 3, Issue 1 (March 1991), pp. 67–99.

[6] J.C. Palencia and M. González Harbour, "Response Time Analysis for Tasks Scheduled under EDF within Fixed Priorities". Proceedings of the IEEE Real-Time Systems Symposium, Cancun, Mexico, December, 2003, ISBN:0-7695-2044-8, pp. 200,209.

[7] J.L. Medina Pasaje, M. González Harbour, J.M. Drake Moyano, "MAST Real-Time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems". Proceedings of the 22th IEEE Real-Time Systems Symposium, London, UK, December, 2001, ISBN:0-7695-1420-0, pp. 245,256.

[8] M. González Harbour, J.J. Gutiérrez García, J.C. Palencia Gutiérrez, and J.M. Drake Moyano, "MAST: Modeling and Analysis Suite for Real Time Applications". Proceedings of the 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, June, 2001, ISBN:0-7695-1221-6, pp. 125,134.

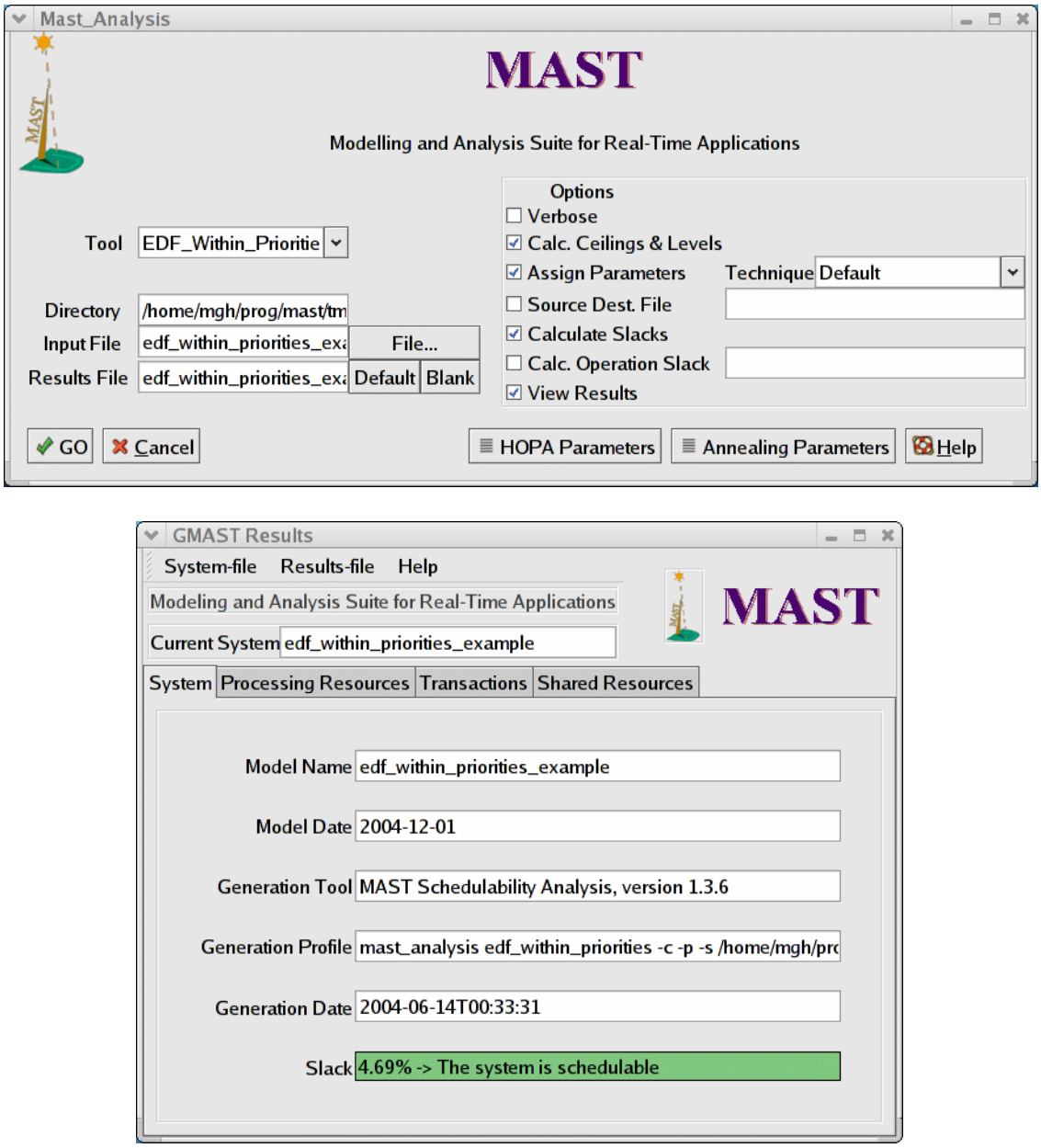

Figure 11. MAST analysis invocation

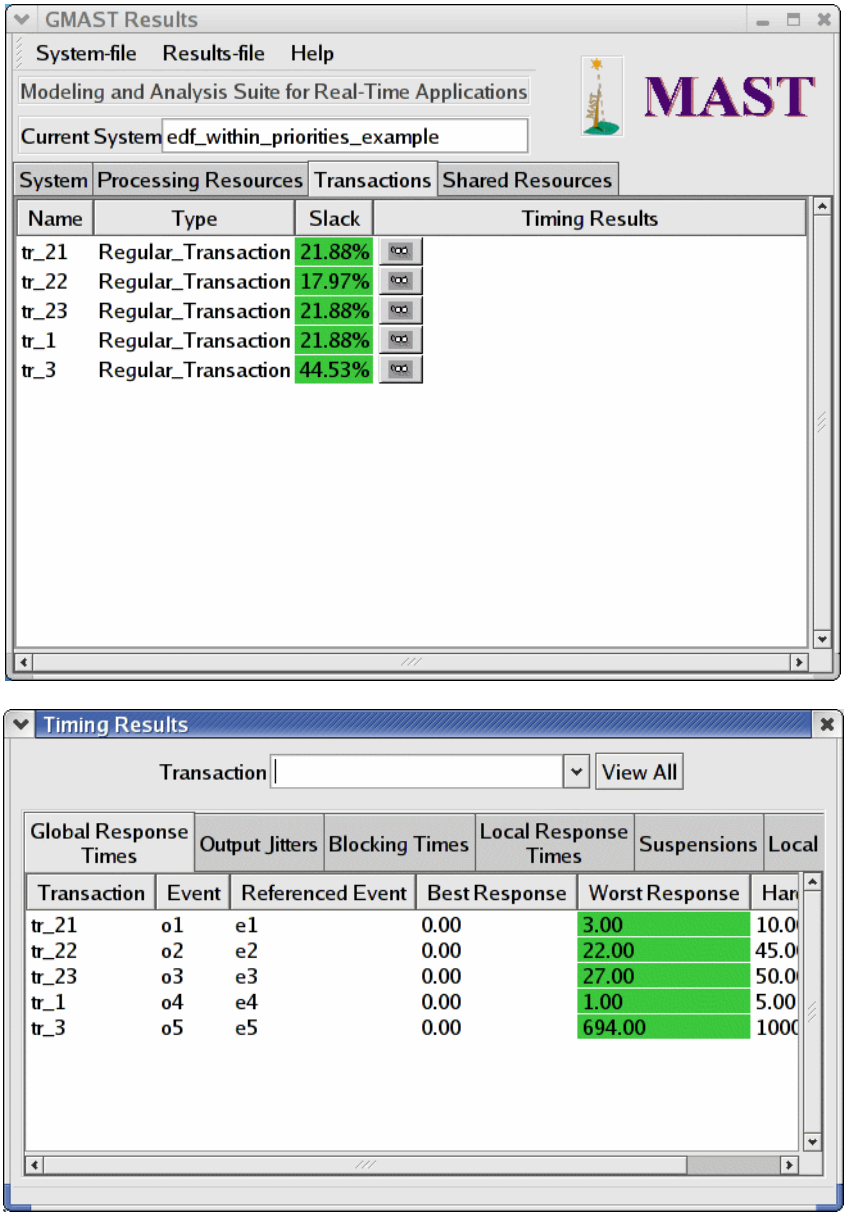

Figure 12. Transaction slacks and timing results