OASIS project: deterministic real-time for safety critical embedded systems

Stéphane Louise, Vincent David, Jean Delcoigne,
and Christophe Aussaguès
CEA-DRT/LIST/DTSI/SLA- Bâtiment 451 - CEA/Saclay
91191 Gif sur Yvette CEDEX - France
(e-mail:firstname.lastname@cea.fr)

15th March 2002

Introduction

Safety critical systems is a growing industrial concern. It is a particular and long time interest for embedded or I&C systems, in nuclear power plant or aircraft applications. Since automotive industry is to use more and more microcontrollers or microprocessors with software in the near future[Bre01], concerns about safety of these systems is becoming mainstream. At the system level, because of intrinsic complexity, it is difficult to guarantee a high dependability. Typical applications should be able to manage numerous control tasks with several different time scales. They does not only demand correctness of algorithms, but also a correct management of tight time constraints, usually dictated by the environment. For system and application design, there is a conflicting interest between algorithm design and time scheduling design. Algorithm design favors a task per targeted control and so a multitasking approach. Time scheduling, on the other hand, has to take care of both strict and precise local chain of events, that needs a careful design in a multitasking environment, and mostly independent events at the system scale, more multitasking prone. One of the highest difficulties arises from the possible existence of very different time scales.

Overview of the OASIS model

The OASIS model comes from a project led by the CEA in association with EDF and Framatome-ANP. Its main goals are to bring a solution to often met issues in the design, implementation and qualification of safety critical embedded systems. One of the main goal of the OASIS project is to allow the programmer to focus on algorithms. Simple statements of time constraints for each task environment -called agent- are enough to guarantee timeliness in the execution, as long as the system sizing is correct for the application. The key points, as shall be seen, is time and scheduling full determinism both in processing start times and deadlines, and in the communication mechanisms. It also allows a full determinism for the application behavior both when system is in nominal functioning and when it is in anomalous functioning thanks to tasks confining. A further consequence is to enable designers to check that the system fits its maximum workload, and also to demonstrate the system safety.

Functions of general I&C systems are data acquisition, processing and operating control devices. Classical system design generally leads to artificially break apart some of these aspects of a single action on a system. Secondly it leads to a multiplication of the number of tasks and communications among them. As a consequence, system worst case load is harder to compute and the application harder to design. This also leads to a higher cost of the system due to bad sizing, to a higher design and implementation cost, and complicates check and validation phases.

In the OASIS model, an application is defined as a set of a known finite number of agents (tasks) enabled to communicate if need be, and cooperating in order to reach their objectives. Each agent is an autonomous running entity composed of a determined number of elementary processing acts. These acts can be chained, depending on internal or external conditions to any agent. The whole model is based on a Time Triggered Approach[Kop] which means that commutations are triggered off by timer interrupts. Each basic processing is associated with an interval of time for its execution, determined by time related information embedded in agent program code.

Time and communication in the OASIS model

The basic mechanism for time management with OASIS is modelized with the "ADV" (advance) instruction. Let's show the following example:

{processing_1; ADV(m); processing_2; ADV(n);}

this means that $start\_2 = agent\_start\_date + m$ is the latest end date of *processing_1* and also the earliest start date of *processing_2*. $end\_2 = start\_2 + n$ is the the latest end date for *processing_2*, then completely defining the interval of time for *processing_2*. As a consequence, the OASIS micro-kernel ensures that *processing_2* cannot start before date $start\_2$, that *processing_2* shall be run between $start\_2$ and end_2 dates. Moreover, if its execution is not finished before end_2, it stops *processing_2* then forks execution for error processing. It should be noticed that for determinism sake, the actual date, time and system state seen by any processing correspond to its earliest start date. Moreover, each agent can be associated with a different default clock so that it is easier for the programmer to manage different time scales, but all time statements can be associated with another agent clock if need be.

The OASIS model enables to use such time constraints with any control flow statements, so that elaborate processing acts chaining can be implemented, as shall be shown in the full paper.

The other important mechanism for parallel programs like OASIS applications is the availability of some kind of information sharing. There are two such mechanisms in the OASIS model. The first one is temporal variables, real time data flows. This is an information that can be shared implicitly between agents. Past values of this time stamped data flow can be read by any agent that needs it. The second mechanism for data communication is explicit message passing. Any processing of an agent can send a message to another agent, at a given date in the future. As a consequence this mechanism can impose further restrictions on interval of time available for a processing execution, but it is also taken in charge by the OASIS model and tools. It also avoids by construction any underflow for these kinds of information sharing between agents. Both mechanisms shall receive further explanations in the full paper.

Timeliness

Timeliness is the ability to execute all planned activities in a timely manner. This requires to demonstrate three properties: no processing is delayed or omitted, no processing is started too early (*i.e.*

before its earliest start date) and all processing meet their deadlines. The first two properties are verified by construction in the OASIS model and runtime environment[DAD00], as shall be clearly demonstrated in the full paper. This is a simple consequence both of the model and of the timing constraints included in the program code of each agent. Since the OASIS model is a Time Triggered Approach, a simple time-driven scheduling at runtime enables to perfectly manage processing activation and early detection of any anomalous, time related behavior of the application. As a consequence, no asynchrony can impact these basic properties of the OASIS model. The third property is then tied to the schedulability of the application and as shown in the following, to system sizing or computing power.

Sizing and schedulability

Schedulability and ability to meet every processing deadline is tied to the knowledge of an actual upper bound to all processing computing time in the worst case. Knowing the exact worst case execution time is not a necessity for schedulability. Knowing an upper bound (even large) is enough to ensure that all processing can be finished on time by using a deadline driven scheduler to trigger off tasks activation on the target systems. This technique is efficient, rigorous and optimal if programming model is adequate, like the OASIS model. With the knowledge of these upper bounds it is possible to be sure that the target system has enough computing power to run the application in the worst case. Other related works has shown how to statically compute the worst case workload of a target system that use a dynamic deadline driven scheduler. What is noticeable by comparison with other models is that the sizing issues are a completely separated matter from the application design issues (and in particular, time design issues).

Of course, knowing tighter upper bound to worst case execution time for each processing enables also a tighter evaluation of computing power needed and thus a better evaluation of system sizing. This can allow a cost reduction of the target system and is a special interest for industrial applications. That is why, although this issue is not directly related to the OASIS model, the evaluation of worst case execution time is also an interest of the OASIS project team.

The OASIS development tools

All necessary tools for implementation of the OASIS model for an embedded application, have been developed and are already fully operational. The code production chain tool include a compiler for a semi-formal language ΨC implementing the basic OASIS mechanisms (ADV, temporal variables and message communication) where purely algorithmic parts are written in $ANSI - C$. The compiler parses and checks consistency of the code as it generates a neutral code. The whole code generation also generates a complete task interface runtime, compiles and links all the code to obtain the complete application. The linking stage also generates static memory tables to protect the whole application and the runtime so that determinism is still guaranteed when a particular agent make anomalous memory accesses. The temporal dependencies are calculated on the state-transition diagram and the buffers sizing (*e.g.* for temporal variables and messages queues) are automatically performed too. It also performs graph generation for system safety analysis and check and sizing checks.

At present, the kernel is already available for Motorola m68k target systems, and studies and development began for Intel IA32 targets. Two industrial transfers are already under way.

Safety Oriented Embedded runtime

In order to achieve run time safety, it is a necessity to use all possible available mechanisms in a computer system. This has been done in the OASIS system. Indeed, privileged mode of processors limited only to critical run-time sections and memory protection between agents is achieved through MMU. At run-time, compilation deduced criterion like correct processing chaining and commutation or deadline checks are performed by the OASIS runtime environment without time penalty, thus enabling to accurately ensure that all agents behaviors are nominal and that any faulting task can not have any incidence on the other. From design to execution of the application, all possible mechanism allowing an early fault detection are implemented so that the deterministic behavior of an OASIS application is also ensured when anomalous behavior appears within agents. It also ensures data coherency and copies messages or data flows between agents. These are the specific tasks of the micro-kernel. This micro-kernel, whose specifications shall be shown in the full paper, is not a classical real-time monitor but simply execute and check control graphs. This micro-kernel is dedicated to this only task and is application independent and so is its validation. It is also aimed to be as portable as can be (mostly programmed in ANSI-C and as few assembly as possible).

The modularized approach of OASIS applications and their determinism in execution (so that the same causes induce the same effects) enables both system safety thanks to tasks confining, and a possible incremental per agent based design of the system, thus allowing easier and quicker design, implementation and checking phases.

Conclusion

This approach ensures strong safety properties like data coherency, a real-time unvarying and deterministic behavior, as well as fault detection and confining mechanisms. Determinism is a master key and a necessity in order to perform relevant tests. OASIS enables to reach such an objective with the further bonus of an easier design, implementation, testing phases and validation of a real-time system. It solves lots of classical issues encountered in real-time system design and programming. It implements a number of innovating features making a quantum leap in real-time methods. All tools are already fully operational and the project has a great vitality thanks to a number of related work, in order to take advantage of distributed systems, their synchronization or in another scope to compute tight evaluations of Worst Case Execution Time with cache memories on a multitasking system. As a new OS approach for embedded real-time applications, it solves a number of issues of primary importance for highly safety sensitive applications, like 1E/A systems in nuclear power plants which was the OASIS original target. Nonetheless, its intrinsic qualities make it become a sensible choice also for mainstream safety oriented applications.

References

[Bre01] E. Bretz. By-wire cars turn the corner. *IEEE Spectrum*, pages 68–73, April 2001.

[DAD00] Vincent David, Christophe Aussaguès, and Jean Delcoigne. Seeking a deterministic multitask framework for safety critical systems: the oasis approach. In *ANS/ENS, International Topical Meeting on Nuclear Plant Instrumentation*, 2000.

[Kop] H. Kopetz. The time-triggered approach to real-time system design. In SpringerVerlang, editor, *Predictably Dependable Computing Systems, ESPRIT basic research series*.