Orccad, a framework for safe robot control design and implementation

Daniel Simon, Roger Pissard-Gibollet, Soraya Arias

ORCCAD, a framework for safe robot control design and implementation

Daniel Simon, Roger Pissard-Gibollet and Soraya Arias
INRIA Rhône-Alpes, 655 avenue de l'Europe
38330 MONTBONNOT ST MARTIN, FRANCE
http://sed.inrialpes.fr/Orccad/

Abstract

Robotic systems are typical examples of hybrid systems where continuous time aspects, related to control laws, must be carefully merged with discrete-time aspects related to control switches and exception handling. These two aspects interact in real-time to ensure an efficient nominal behaviour of the system together with safe and graceful degradation otherwise. In a mixed synchronous/asynchronous approach, ranging from user's requirements to run-time code, Orccad provides formalised real-time control structures, the coordination of which is specified using the ESTEREL synchronous language. CAD tools have been developed and integrated to help the users along the steps of the design, verification, implementation and exploitation processes.

1 Motivation

A fully fledged robotic system includes various subsystems coming from various fields of science and technology like mechanical engineering, automatic control, data processing and computer science. The goal of the control architecture is to organise coherently all these subsystems so that the global system behaves in an efficient and reliable way to match the end-user's requirements.

Robotics primarily deals with physical devices like arms or vehicles. These devices are governed by the laws of physics and mechanics. Compared with virtual, purely computing systems, they exhibit inertia and their models are never perfectly known. Usually their behaviour can be described by differential equations where time is a continuous variable. Their state can be measured using sensors of various kind which themselves are not perfect. Control theory provides a large set of methods and algorithms to govern their basic behaviour through closed-loop control ensuring the respect of required performance and crucial properties like stability.

Robots of any type interact with their physical environment. Although this environment can be sensed by exteroceptive sensors like cameras or sonars, it is only partially known and can evolve through robot actions or external causes. Thus a robot will face different situations during the course of a mission and must react to perceived events by changing its behaviour according to corrective actions. These abrupt changes in the system's behaviour are relevant of the theory of Discrete Events Systems.

Besides the logical correctness of computations the efficiency and reliability of the system relies on many temporal constraints. The performance of control laws strongly depend on the respect of sampling rates and computing latencies. Corrective actions must be usually executed within a maximum delay to insure the mission success and the system's security. The optimisation of computing resources is still a relevant problem especially for remote autonomous robots like planet rovers[29] or underwater vehicles[31] where both on-board space and energy are severely limited.

Therefore robotic systems belongs to the class of hybrid reactive and real-time systems in which different features need to use different methods and tools to be programmed and controlled. The ORCCAD[5] environment is aimed to provide users with a set of coherent structures and tools to develop, validate and encode robotic applications in this framework.

2 The Orccad architecture

2.1 Basic statements

The formal definition of a robotic action is a key point in the ORCCAD framework. It is based on the following basic statements :

- In many cases the physical tasks to be achieved by robots can be stated as automatic control problems, which can be efficiently solved in real-time by using adequate feedback control loops. Let us mention that the *Task-Function* approach[33] was specifically developed for robotic systems;
- The characterisation of the physical action is not sufficient for fully defining a robotic action : starting and stopping times must be considered, as well as reactions to significant events observed during the task execution; this will be further used to design complex missions through actions composition.
- Since the overall performance of the system relies on the existence of efficient real-time mechanisms at the execution level, particular attention must be paid to their specification and their verification.

From the users and programmers point of view, the specification of a robotic application must be modular, structured and accessible to users with different domain of expertize. The *end-user* concerned with a particular application must be provided with high level formalisms allowing to focus on mission specification and verification issues; the *control systems engineer* needs an environment with efficient design, programming and simulation tools to express the control laws which then are encapsulated for the end-user.

2.2 The Orccad approach

In ORCCAD, two entities are defined in order to capture the aforementioned requirements. The **Robot-Task** (RT) models basic robotic actions where control aspects are predominants. For example, let us cite hybrid position/force control of a robot arm, visual servoing of a mobile robot following a wall or constant altitude survey of the sea floor by an underwater vehicle. The RT characterises in a structured way closed loop control laws, along with their temporal features related to implementation and the management of associated events. These events are :

- preconditions, which can be associated with measurements and watchdogs
- events and exceptions of three types :
  - synchronisation signals reports the occurence of a particular state in a RT, they can be used to synchronise different RTS;
  - type 1 exceptions are locally processed in the RT, e.g. by tuning a parameter of the control law;
  - type 2 exceptions signal that the RT cannot run to completion and must be switched for a new one, chosen by the supervision controller;
  - type 3 exceptions are fatal for the application and must, as far as possible, drive the system in a safe recovery state.
- postconditions are emitted when the RT successfully terminates.

For the mission designer this set of signals and associated behaviours represents the *abstract* view of the RT, hiding all specification and implementation details of the control laws (Figure 1). The characterisation of the interface of a RT with its environment in a clear way, using typed input/output events, allows the composition of them in an easy way in order to construct more complex actions, the **Robot-Procedures** (RPs) : The RP paradigm is used to logically and hierarchically compose RTs and RPs in structures of increasing complexity. Usually, basic RPs are designed to fulfil a basic goal through several potential solutions, e.g, a mobile robot can follow a wall using predefined motion planning, visual servoing, or acoustic servoing according to sensory data availability. RPs design is hierarchical so that common structures and programming tools can be used from basic actions up to a full mission specification.

These well defined structures associated with synchronous composition, thanks to the use of the ESTEREL language [4], allows for the systematisation and thus the automatisation of the formal verification on the expected controller behaviour. This is also a key to design automatic code generators and partially automated verification. Formal definitions of RPs and RTs together with associated available formal verification methods may be found in [24].

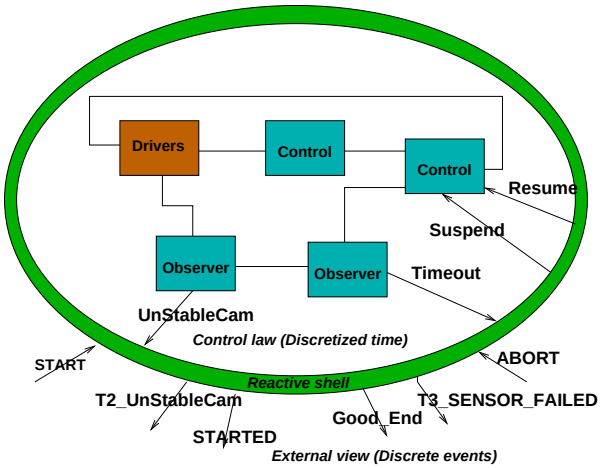


Figure 1: Encapsulation of the control law in a reactive shell

3 Specification of *Robot-Tasks*

The design of the robot controller begins with the design and test of basic actions when the necessary actions do not pre-exist in the RT library. Let us now illustrate the design and validation process of a RT through an example of underwater robotics taken from [38], where the goal of the KeepStableCam RT consists of the stabilisation of an underwater vehicle using visual servoing.

Modular Specification in Continuous Time In fact, it should be emphasised that the RT designer may select easily the adequate models and tuning parameters in ORCCAD, since they belong to some predefined classes in an object-oriented description of the control available through the graphical interface depicted by figure 2. The control algorithm is designed as a block-diagram, where elementary *algorithmic* modules are connected through input/output ports, e.g. `StableCam` in Figure 2a. The value of parameters must be set to instantiate the design. It is also necessary to enter the localisation of data processing codes (C files).

The particular *Physical-Resource* module (`VORTEX` here) provides a gateway towards the physical system through its Driver ports. A complete description of more complex RTs and of sensor-based tasks may be found in [38].

Figure 2: a)Functional and temporal attributes of a Module – b) Specification of the reactive behaviour of a RT

Timed-Constrained Specification The resulting specification defines the action from a continuous time point of view, i.e independently of time discretization and other implementation related aspects which are considered in a next design step. The passage from the above continuous time specification to a description taking into account implementation aspects is done by associating temporal properties to modules, i.e. sampling periods, assumed worst case execution time (WCET), communications and synchronisations between the processes.

Modules can be synchronised in several ways :

- explicitly periodic modules are connected to a system clock through a hidden input port ;
- an input port can be synchronised on an output port of another module, thus inheriting its period ;
- it can be synchronised on an external source, e.g. an interrupt coming from a vision processing system.

In default mode, for basic control purpose, the designer specifies a single-loop, single-rate controller where all the algorithmic modules are gathered in a single real-time task for which a single sampling period is defined.

More complex timing behaviours, e.g. multi-rate control, make use of more or less tight synchronisations between modules and clocks. In many cases simple design rules can be used to get a correct (deadlock free) timing diagram [37]. For more complex designs the synchronisation and timing diagram can be built interactivelly and a Petri net model of the synchronisation skeleton of the RT is automatically extracted from the timed block-diagram. Thus its logical correctness and timing analysis (using the provided WCET of modules) can be done using the underlying model in the (max,plus) algebra [36]. Finally feedback scheduling can be specified and implemented provided that some additionnal instrumentation (for variable clocks generation and accurate online execution time measurement) has been included in the real-time platform ([40]).

Specification of the reactive behaviour The logical behaviour of the RT is automatically encoded in ESTEREL through a graphical window (Figure 2c) (here we specify that the `UnStableCam` signal must be treated as a type 2 exception). Thanks to the strong typing of events and exceptions the code generator was proved to be correct and thus guarantees that crucial properties like safety and liveness are true [24]. Besides user's defined signals (pre and post-conditions, exceptions), the code generator builds a large ESTEREL file where hidden system signals are also declared. These signals will be used at run time to spawn, suspend or resume all the real-time threads necessary for the execution of the RT.

4 Design and Analysis of *Procedures*

After all necessary basic actions have been designed and validated, the user now wants to use them in more complex procedures to perform a useful underwater inspection mission [38]. Here is the detailed specification of the KEEPSTABLE procedure in order to enlighten the specification and analysis process proposed for basic actions composition.

4.1 Synchronous programming

However, even if these tools provide efficient run-time code they remain difficult to be directly used. They require expertise from designers and programmers who, as a consequence, spend time in programming tricks rather than being concentrated on the specification and validation of actions and applications. Therefore, more friendly CAD systems, built over basic tools like C, ESTEREL or VxWorks, must be provided to improve both programmer's efficiency and programs reliability.

From the verification side, the compositionally principle must preserve the coherence between underlying mathematical models, in order to be able to perform formal computations at any level. As an example, the use of the single ESTEREL [4] synchronous reactive language as a target for automatic translation is a way of preserving a logical structure whatever the complexity of the application.

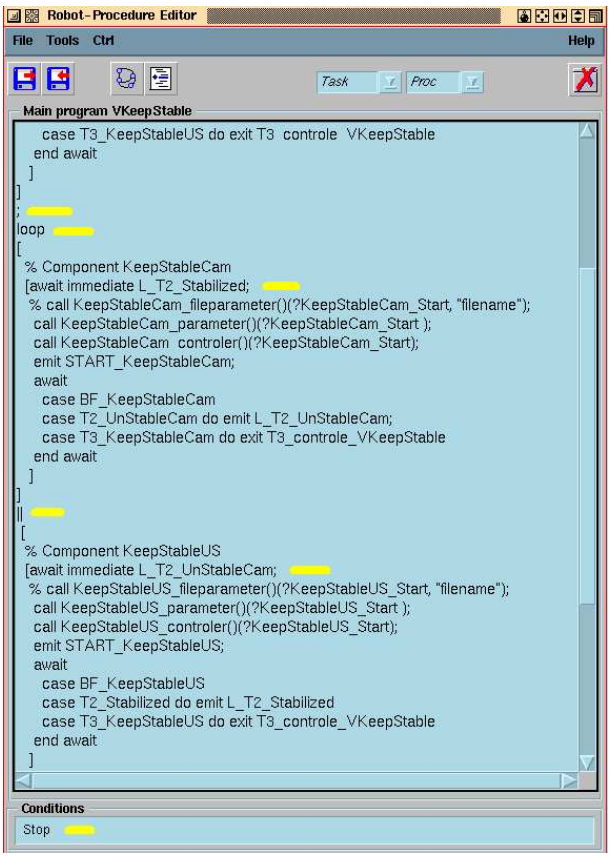
A consequence of this point of view is that the basic entities have to be carefully studied and also that composition operators should have a proper semantics.

The formal definition of a robotic action is a key point in the ORCCAD approach. The *Robot-Task* (RT) models basic robotic actions where control aspects are predominants, like the hybrid position/force control of a robot arm or the visual servoing of a mobile robot. The characterisation of the interface of a RT with its environment through typed input/output events allows to compose them easily in order to construct more complex actions, the so called *Robot-procedures* (RP), while hiding most implementation details. In its simplest expression, a RP coincides with a RT, while the most complex one might represent an overall mission. Briefly speaking, it specifies in a structured way a logical

and temporal arrangement of RTs in order to achieve an objective in a context-dependent and reliable way, providing predefined corrective actions in the case of unsuccessful execution of RTs.

These formally defined structures[26] associated with synchronous composition, thanks to the use of the ESTEREL language, allows to systematise and therefore to automatise formal verification about the expected controller behaviour. In complement of two other features of ORCCAD i.e. the object-oriented model and the possibility of automatic code generation, this capability of verification is a key point for meeting the requirements of safety in the programming of critical applications.

4.2 Procedure specification




```
do
  SEQ(do
      KeepStableUS
    until Stabilized
    ;
    loop
      PAR(when T2_Stabilized when T2_UnsStableCam
          do                 do
            KeepStableCam       KeepStableUS
          until UnStableCam   until Stabilized)
    end loop
     )
until Stop
```

Figure 3: Specification of a Procedure using : a) the GUI – b) the MaestRo language

The KEEPSTABLE procedure aims at stabilising the underwater vehicle in spite of perturbations. The vehicle can be stabilised in two ways : it can be remotely locked on its target using either visual servoing or acoustic sensors. Here visual servoing (running the KeepStableCam RT) is considered as the nominal mode. However, if the camera loses the target, the controller must switch to sounders stabilisation (degraded mode) running the KeepStableUS RT until being able to recover the vision tracking mode. This redundancy is useful to increase the safety and efficiency of the system and such a situation where several RTs are exceptions of each other is a typical structure in our procedures. Once again, the ESTEREL language is used to specify that actions are run in sequence or in parallel, or must preempt each other. Thanks to the structure of ORCCAD programs, the specification of this procedure can be written in two ways which both avoid the end-user to write himself the full source ESTEREL file :

- The Robot-Procedure Editor displays the external view of selected RTs and RPs. The user just has to add some statements to express, e.g., sequencing (;), parallelism (||), or escape mechanisms (trap-exit) to complete the specification (these additional statements are highlighted in Figure 3a). As the designer only access the external views, he cannot jeopardise the properties of the previously defined actions.

- The MAESTRO language [10] has been developed to target ESTEREL (Figure 3b). This language has been designed to be more "natural" for robotic practitioners. At compile time it expands in ESTEREL statements and also performs some preliminary verifications like liveness.

In this example the alternance between the two RTs is specified inside a parallel statement (PAR), in which each RT is guarded by a T2 exception emitted by the other RT. As these exceptions are mutually exclusive the two RTs cannot run simultaneously : anyway this property will be formally verified in the next section.

At compile time the control code of this procedure is translated into an automaton which output functions are automatically filled with calls to the underlying RTOS, thus alleviating the burden of the programmer while preserving formal verification capabilities.

Robotic applications are built incrementally by adding new RTs and RPs which behaviour, e.g. synchronisation, are encoded in the same way. The next example describes a RP used to synchronise the motions of the underwater vehicle and of its associated arm.

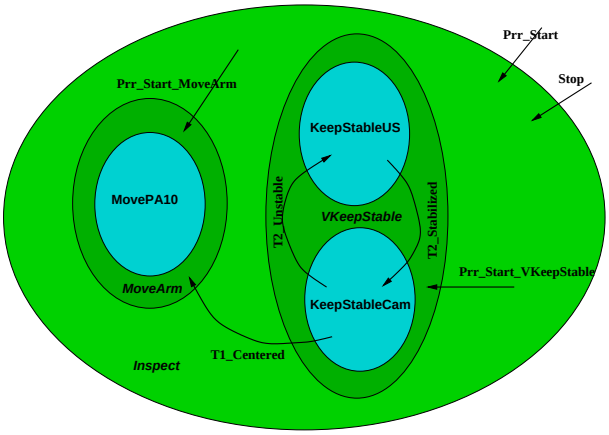


Figure 4: Structure of a RP synchronising the arm and the vehicle

4.3 Logical Behaviour Verification

First, the satisfaction of *crucial properties* can be checked. Concerning the *safety property* (any fatal exception must always be correctly handled) the process is as follows : knowing the user's specification defining the fatal exceptions and the associated processing, a criterion is automatically built to define an abstract action. The abstraction of the global procedure automaton with respect to this criterion is then computed. The absence of the "Error" action in the resulting automaton proves that the safety property is verified.

The *liveness property* (the RP always reaches its goal in a nominal execution) is proved in a similar way. The "Success" signal is emitted at the end of all successful achievements of a RP. The abstraction of the procedure automaton crossed with an adequate criterion built with this signal must be equivalent by bi-simulation to a one state automaton with a single action, the "Success" one.

Conflicts detection : We are interested here in checking that during the RP evolution, there does not exist instants where two different RTs are competing for using a same resource of the system. Here, the physical resource controlled by the RTs (the underwater vehicle) is considered as well as the software resources used by the controllers (real-time tasks). For example, one wants to verify that the RTs KEEPSTABLECAMERA and KEEPSTABLEUS never compete to apply different desired force inputs to the vehicle thrusters during all the RP evolution. The global automaton is reduced to the only interesting signals `Activate...` and `CmdStopOK...`. Thus by clicking on the conflicts button one can check that these two signals alternate during the RP life insuring that a control law can be started only after confirmation that the previous one is stopped and that the transition is as fast as possible to ensure the system's stability (Figure 5).

Finally, the *conformity of the RP behaviour with respect to the requirements* is verified. For example, we want to certify that the vision-servoing and acoustic-servoing RTs can alternate for an arbitrary number of time during the life of the stabilisation procedure. This property can be checked by observing the abstract view given by the ORCCAD graphical interface (figure 6) : after minimisation and abstraction through behavioural bi-simulation ([6], the original automaton

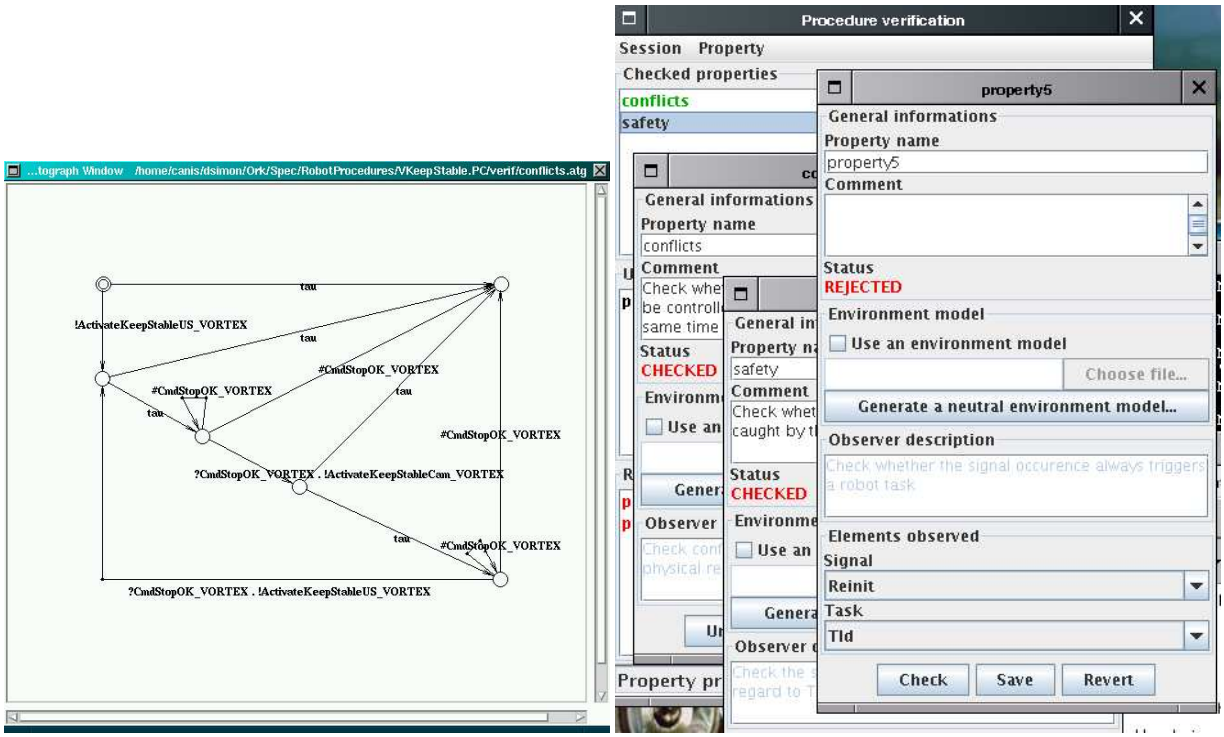


Figure 5: Checking for conflicts a) using explicit automata b) using observers

(which has 17 states and 168 transitions) becomes small enough to be visually analysed. As expected, the procedure begins with the `KeepStableUS` RT (arc 1) : then, the two RTs altern in a loop, following arcs 2 and 3. The occurrence of the `Stop` signal is the only way to exit the procedure (arcs 4 and 5). The user just have to click on the name of signals which are relevant for the property to be checked to launch the verification process.

5 Implementation

5.1 Switching mechanism

Raising a type 2 exception or a ”well-finished” signal request the embedding RP to smoothly stop the running RT and starting the next one, according to its own recovery program. Switching between two RTs must insure that :

- The control laws are not conflicting, i.e. the degrees of freedom of the system are all controlled, and only once.
- The boundary conditions of the successive control laws must be compliant, in some cases it would be necessary to insert a special purpose transition RT to gain a full control of the transition process [34];
- The configuration of the set of real-time tasks is valid before starting the new control law.
- The switching time must be small w.r.t. the plant’s time constants. Ideally, the delay during which the system is not controlled should be in the range of one sampling period.

The steps of the switching mechanism have been identified as follow :

- Upon reception of a type 2 exception or postcondition from the running RT, choice of the next one to start, starting of the transition phase;
- Initialisation of the new real-time tasks and instantiation of the communication ports. According to the load of the processor, it may be necessary to decrease the sampling period of the running RT (degraded mode allowing to keep the system under control);
- After fulfilment of the preconditions of the second RT, stopping the first control law and starting the second one. This delay must be minimised to fit in one sampling period;

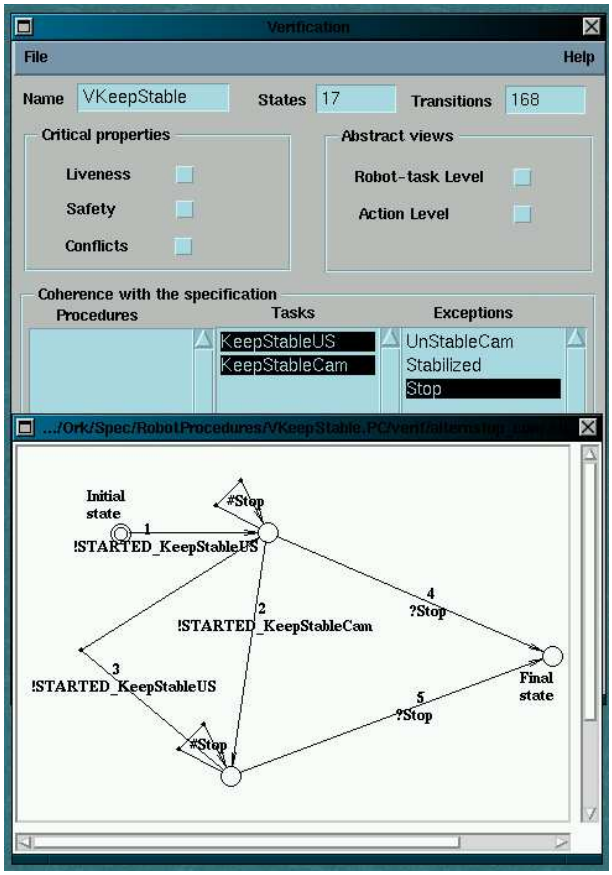


Figure 6: An abstract view of KeepStable

- Suspension or destruction of the former real-time tasks

In practice it appears that such a complex mechanism is very difficult to set up and port on different targets. Indeed, according to the properties of the used control laws and plant, it is always not necessary to use the fully-fledged smooth transition protocol ; in particular a simpler transition procedure has been implemented and succesfully experimented in the framework of vision based tasks ([2]).

5.2 Design and code generation toolset

Figure 7 pictures the compilation process and integrated tools. The ORCCAD structures are instantiated as C++ classes (or C structures) using virtual system calls, provided by the kernel. For a single processor implementation, all ESTEREL files corresponding to the RTs and RPs logical behaviour are gathered (i.e. put in parallel) in a single file which can be further compiled into various formats. The SC format (sorted boolean equations) is post-processed in C and encapsulated in a high priority real-time thread.

The application is finally compiled and linked with run-time libraries to make the down-loadable code. Currently ORCCAD targets VxWorks and RTAI as hard real-time systems and Solaris and Linux as soft real-time systems (using Posix threads). Note that building a real-time simulator running on the same target can be easily done using the same code generation toolset : this is simply done by calling a numerical integrator (itself calling a model of the robotic system) in the drivers of the Physical Ressource [39]. However running the full simulation in real-time requires using an oversized processor to get enough computing power. Various dedicated run-time interfaces are also provided to monitor the execution of the controller.

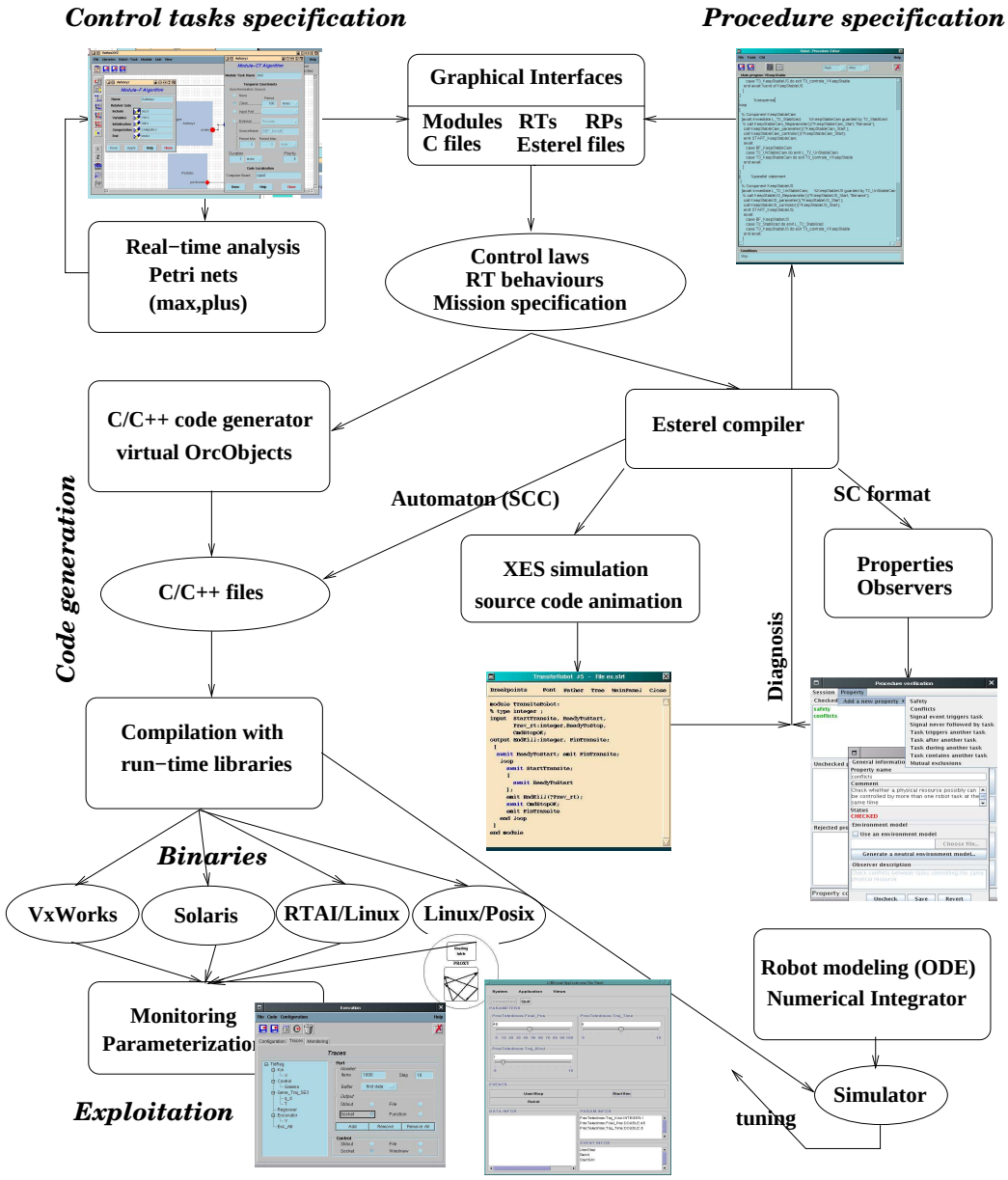


Figure 7: Code generation and associated tools

5.3 Real-time structures

At compile time, the ESTEREL source file is translated into an automaton encoded in C. Input and output functions are associated to allow the automaton to receive and emit signals. These functions are used to interface the synchronous reactive program with the asynchronous execution environment, i.e. the operating system. As the compiler knows nothing about the environment, the output functions are empty and must be filled by the user to make the program effective. Moreover, as ESTEREL is unable to do numerical computations, all calculations related to the execution of control algorithms are called as external procedures. Thus it is necessary to design an *execution machine* [1], the general structure of which is given by Figure 8a. Signals are collected to build the current event which is given to the automaton. Output actions calling extern calculation procedures must be carefully interfaced with the RTOS. Writing by hand this execution machine would be tricky and error prone.

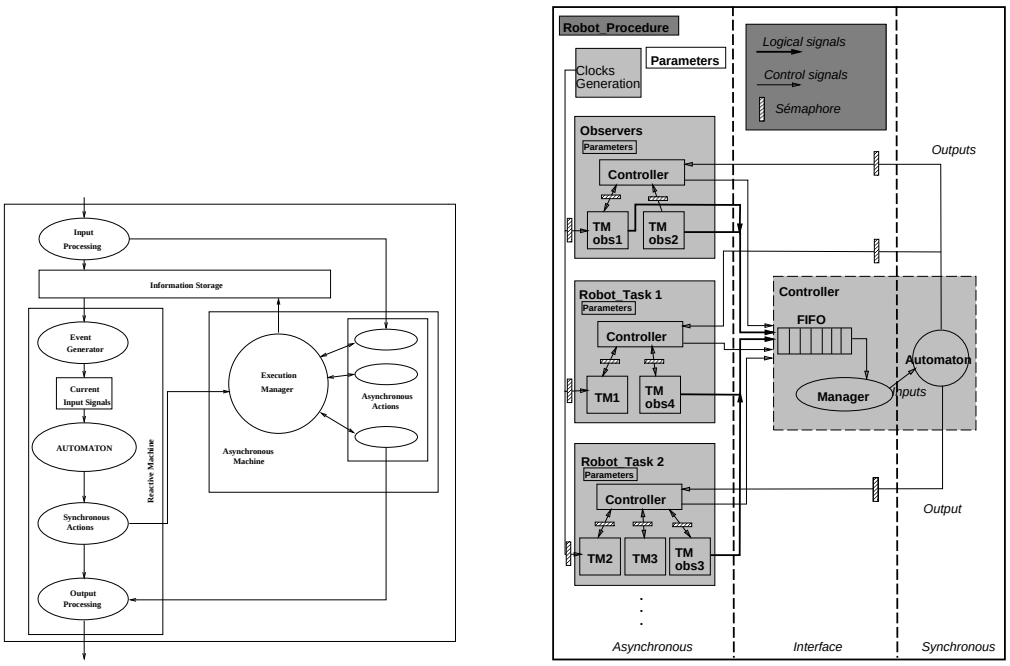


Figure 8: a) Structure and b) Implementation of the execution machine

Thanks to the ORCCAD structures this machine can be automatically generated by the system (Figure 8b). A main "system" task sets up the whole system and in particular generates the needed clocks used to trigger the periodic calculation modules. Real-time threads are made periodic by blocking their first input port on a semaphore which is released by clock ticks.

The automaton is the highest priority task : it is awakened by the occurrence of input signals related to pre-conditions, exceptions, and post-conditions. In reaction, it tells the RTOS what modules must be spawned, resumed or suspended. Although this automaton is crucial for a safe and successful behaviour of the application, it spends most of time waiting for input events during the periodic execution of control algorithms managed by the RTOS. Typically, a transition of the automaton takes a few microseconds while the duration of robotic actions may range from seconds (e.g. manipulation tasks for a robot arm) to hours (e.g. mapping mission for an autonomous underwater vehicle).

6 Related work

Many robotics controllers software inspired from Component-Based Software engineering have been developped (Claraty [30], Cosarc [32], Oscar [22], Orca [7], Orocos [8], HRPOpen [23]). All these approches are done by research teams, often dedicated to a single robot type (mobile robot, humanoid robot or robotic arm) and are not adopted by a big community. To tackle the control robotics software complexity, it is mandatory to pool resources and efforts. However although ORCCAD has been primarily developed and used for robot control purpose it belongs to a more general family of model driven design methods and tools for computer controlled systems [11] [15].

Some other layered approaches in robot control show architectural similarities with our. In the fields of synchronous programming applied to robotics let us cite [28] where data and control flows are gathered using the SIGNAL synchronous data-flow language. Using the same formalism for numerical calculations and tasking management allows for checking the temporal coherency of data in the whole application. However, if SIGNAL is well suited to specify signal processing algorithms, long computations like computing the explicit dynamics of a robot arm should be done by external functions written in a host language like C.

ControlShell (now Constellation) [35] proposes an architecture quite similar to ORCCAD. At the level of control actions it allows for the construction of models of components such as PIDs or trajectory generators which can be gathered through a block-diagram oriented GUI. Event-driven aspects are handled by hand-made (and thus quite simple) Finite State Machines for which formal verification methods are not provided.

Besides this commercial offer, the OROCOS project's aim is to build open source software for robot control, from the lowest, real-time servo control, over intelligent sensor processing, to high-level human-machine interfacing. The emphasis is on building components, not on designing architectures [8]. However the proposed framework, ranging from control design to real-time code generation, also enforces the separation of data flow (periodic control laws computation) and execution flow controlled via Finite State Machines.

7 Summary and perspectives

In this paper we have described the ORCCAD approach to formalize control structures in controllers for robotics and others embedded systems. In this approach, the Robot-Task, where a discretized control law is encapsulated in a logical behavior, is a hybrid structure at the frontier between continuous and discrete time aspects. The formalization of these structures allows for the formal verification of programs and automatic down-loadable code generation.

In such control systems, the interleaving of events and activities in a real life mission accounts for our hierarchical design and verification process, where complex actions are designed from already validated basic blocks lying at a lower level. The ESTEREL synchronous language was found to be effective to specify and encode the coordination of actions, from the low level logical management of real-time tasks up to the application layer. To further ease the specification process, ORCCAD provides friendly user interfaces:

- Starting from the specifications given through the ORCCAD GUI, the user is provided with control oriented pre-defined structures. Automatic code generation allows him to take advantage of real-time tools and synchronous languages with a moderate programming effort. These structures are also a key for a partial automatization of the verification process.
- The ESTEREL compiler does not provide the interface functions with the environment. To avoid messy and error prone low level interfacing work, ORCCAD automatically generates the execution machine managing both the synchronous automaton and the asynchronous real-time threads. Thus the user can concentrate on application specification and validation rather than low level programming tricks.
- However, compiling reactive programs into explicit automata is subject to size explosion, and behavioral verification through visual inspection of automata becomes impossible for industrial size problems, even after reduction and bisimulation. An ongoing work consists of the identification of classes of relevant properties to be encoded in generic observers : after parameterisation these properties can be checked by the compiler itself (e.g. figure 5b). A dual approach consists in using a discrete events controller synthesis algorithm to build a safe-by-construct controller from the desirable properties, e.g. from a fault tolerance specification.

ORCCAD has been used in laboratory applications such as programming the BIP 2000 biped control laws [3], real-time simulation of a teleoperation platform [39], high level programming of an underwater robotic system [25] and feedback scheduling experiments [40].

The Orccad concepts and tools have been selected to support the development of a set of ESA projects. In particular, in the MUROCO (Multiple Robot Controller [18]) project the Orccad structuring of Events, Actions and Tasks, as well as the utilisation of Esterel as a mission specification language are used in a dedicated tool developed by TRASYS Space. Up to 63 actions are composed in twelve tasks to specify the full ExoMars mission ranging from 'deployment' to 'travelling' and 'experiment cycle' activities. In the VIMANCO on-going project [19], TRASYS in cooperation with INRIA and KUL aims at improving the autonomy, safety and robustness of space robotics system using vision. The developments include the implementation of a Vision Software Library allowing Vision Control for space robots, for which the Orccad controller is foreseen for the real time execution of the vision based tasks.

From a software engineering perspective the Orccad approach allows for the design of robot software controllers using existing theory (Control and Discrete Events theories) and software tools based on existing technologies already used for the embedded and real-time domains (RTOS, Object Oriented languages, Synchronous languages,...). Starting from existing paradigms, our goal is to provide an effective software framework and associated tools well suited for robotic applications and more generally for control and real-time co-design.

We think that the current ORCCADapproach is still valid but that the software tools must be upgraded and constantly updated. The current release has not been re-engineered for eight years, although many partial improvements (e.g. real-time multitasking) have been developed and tested but not cleanly integrated.

Orccad tools must evolve according to the improvements of software engineering such as Model Driven Architecure MDA (e.g. [15] for computer-based control) or Synchronous language [12], [13], [9]). However this necessary evolution can still be based on existing design and control architectures dedicated to the application field.

In the middleware domain, using modern software enginering now lead to the emergence of a large community willing to adopt standards (e.g. see the success of the ObjectWeb consortium [21]) : using such tools have increased drastically software productivity. Among other initiatives the recently created Object Management Group (OMG [16]) robotics corner aims at the elaboration of OMG standards (UML, Corba are OMG standards for example) to ease the integration of robotic systems using modular components.

The work around robotics standardisation must be done in connection with the embedded and real-time community. In particular this community is involved in several collaborative projects to adapt UML and/or MDA standards taking into account performance and real-time constraints (e.g. [27], [20], [17], [14]). Designing a common architecture based on such standards adopted by a large community could be now the cornerstone for efficiently sharing robot control software development and practice.

References

[1] C. André, A. Ressouche, and J.M. Tanzi. Combining special purpose and general purpose languages in real-time programming. In *IEEE Workshop on Programming Languages for Real-Time Industrial Application*, Madrid, 1998.

[2] S. Arias. *Formalisation et intégration en vision par ordinateur temps réel*. PhD thesis, Université de Nice Sophia Antipolis, 1999.

[3] C. Azevedo, N. Andreff, and S. Arias. BIPedal walking:from gait design to experimental analysis. *Mechatronics*, 14/6:639–665, 2004.

[4] G. Berry. The esterel v5 language primer. Technical report, http://www-sop.inria.fr/esterel.org/, 2000.

[5] J.J. Borrelly, E. Coste-Manière, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, and N. Turro. The ORCCAD architecture. *Int. Journal of Robotics Research*, 17(4):338–359, april 1998.

[6] A. Bouali and R. de Simone. Symbolic bisimulation minimisation. In *Computer Aided Verification*, number 663 in Lecture Notes in Computer Science. Springer, 1993.

[7] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback. Towards component-based robotics. In *Intelligent Robots and Systems IROS*, pages 163–168, august 2005. http://orca-robotics.sourceforge.net/.

[8] H. Bruyninckx. Open robot control software: the OROCOS project. In *IEEE International Conference on Robotics and Automation ICRA*, Seoul, may 2001. http://orocos.org/.

[9] J-L Colaço, B. Pagano, and M. Pouzet. A conservative extension of synchronous data-flow with state machines. In *ACM International Conference on Embedded Software (EMSOFT'05)*, Jersey city, New Jersey, september 2005.

[10] E. Coste-Manière and N. Turro. The MAESTRO language and its environment: Specification, validation and control of robotic missions. In *10th IEEE/RSJ International Conference on Intelligent Robots and Systems*, Grenoble, 1997.

[11] J. El-khoury, D. Chen, and M. Törngren. A survey of modelling approaches for embedded computer control systems (version 2.0). Technical Report TRITA - MMK 2003:36,ISSN 1400 -1179, ISRN KTH/MMK/R-03/11-SE, Royal Institute of Technology, KTH, Stockholm, Sweden, 2003.

[12] D. Garlan. Software architecture: a roadmap. In *Conference on the future of Software engineering*, pages 91–101, Limerick, Ireland, june 2000.

[13] D. Garlan and D. Perry. Software architecture: practice, potential, and pitfalls. In *16th International Conference on Software Engineering*, 1994.

[14] S. Gérard, N. Voros, C. Koulamas, and F. Terrier. Efficient system modeling for complex real-time industrial networks using the ACCORD/UML methodology. In *International Workshop on Distributed and Parallel Embedded Systems DIPES*, october 2000.

[15] D. Henriksson, O. Redell, J. El-Khoury, M. Törngren, and K-E. Årzén. Tools for real-time control systems co-design — a survey. Technical Report ISRN LUTFD2/TFRT--7612--SE, Department of Automatic Control, Lund Institute of Technology, Sweden, April 2005.

[16] http://robotics.omg.org/.

[17] http://www.carroll-research.org.

[18] http://www.esa.int/esaMI/Aurora/SEMQA7A5QCE_0.html.

[19] http://www.irisa.fr/lagadic/actions-internationales-fra.html.

[20] http://www.ist-compare.org/.

[21] http://www.objectweb.org/.

[22] http://www.robotics.utexas.edu/rrg/research/oscarv.2/.

[23] F. Kanehiro, H. Hirukawa, and S. Kajita. OpenHRP: Open architecture humanoid robotics platform. *International Journal of Robotics Research*, 23(2):155–165, 2004.

[24] K. Kapellos. *Environnement de programmation des applications robotiques réactives*. PhD thesis, Ecole des Mines de Paris, Sophia Antipolis, 1994.

[25] K. Kapellos, D. Simon, S. Granier, and V. Rigaud. Distributed control of a free-floating underwater manipulation system. In *5th Int. Symp. on Experimental Robotics*, Barcelon, 1997.

[26] K. Kapellos, D. Simon, M. Jourdan, and B. Espiau. Task level specification and formal verification of robotics control systems: state of the art and case study. *Int. Journal of Systems Science*, 30(11):1227–1245, 1999.

[27] F. Loiret and D. Servat. Insights on real time systems architecture modelling for a software engineering viewpoint. In *17th Euromicro Conference on real-time systems ECRTS05*, Palma de Mallorca, june 2005. work in progress session.

[28] Marchand, E., E. Rutten, H. Marchand, and F. Chaumette. Specifying and verifying active vision-based robotic systems with the SIGNAL environment. *Int. Journal of Robotics Research*, 17(4):418–432, 1998.

[29] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous Robots*, 2(4):291–311, 1995.

[30] A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I. Shu, , and D. Apfelbaum. CLARAty: Challenges and steps toward reusable robotic software. *submitted to International Journal of Advanced Robotic Systems*, 2006. http://keuka.jpl.nasa.gov/main/.

[31] A. Pascoal. The AUV MARIUS: Mission scenarios, vehicle design, construction and testing. In *2nd IARP workshop on Underwater Robotics*, Monterey, CA, may 1994.

[32] R. Passama and D. Andreu. COSARC : Component based software architecture of robot controllers. In *1st National Workshop on Control Architecture of Robots: software approaches and issues*, Montpellier, 2006.

[33] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task-Function Approach*. Oxford Science Publications. Clarendon Press, 1991.

[34] A. Santos, B. Espiau, P. Rives, D. Simon, and V. Rigaud. Sensor-based control of holonomic autonomous underwater vehicles. Technical Report 2609, INRIA Research Report, july 1995.

[35] S. Schneider, V. Chen, G. Pardo-Castellote, and H. Wang. Controlshell: A software architecture for complex electromechanical systems. *Int. Journal of Robotics Research*, 17(4):360–380, 1998.

[36] D. Simon and F. Benattar. Design of real-time periodic control systems through synchronisation and fixed priorities. *Int. Journal of Systems Science*, 36(2):57–76, 2005.

[37] D. Simon, E. Castillo, and P. Freedman. Design and analysis of synchronization for real-time closed-loop control in robotics. *IEEE Trans. on Control Systems Technology*, 6(4):445–461, july 1998.

[38] D. Simon, K. Kapellos, and B. Espiau. Control laws, tasks and procedures with ORCCAD: Application to the control of an underwater arm. *Int. Journal of Systems Science*, 17(10):1081–1098, 1998.

[39] D. Simon, M. Personnaz, and R. Horaud. Teledimos telepresence simulation platform for civil work machines : real-time simulation and 3D vision reconstruction. In *IARP Workshop on Advances in Robotics for Mining and Underground Applications*, Brisbane, Australia, october 2000.

[40] D. Simon, D. Robert, and O. Sename. Robust control/scheduling co-design: application to robot control. In *RTAS'05 IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 118–127, San Francisco, March 2005.