Design of Control Procedures for a Free-floating Underwater Manipulation System

D. Simon, K. Kapellos*
INRIA Sophia-Antipolis, B.P. 93
06902 SOPHIA-ANTIPOLIS Cedex, FRANCE

B. Espiau
INRIA Rhone-Alpes, 655 avenue de l'Europe
38330 MONTBONNOT ST MARTIN, FRANCE

Abstract

In this paper we present a preliminary study concerning the stabilization of an underwater free-floating Remotely Operated Vehicle fitted with a redundant manipulator. The controller structure follows the Orccad approach for robotic systems programming, which gathers control laws in continuous time at the low levels and discrete time logical aspects at higher levels. After being designed and verified, the basic actions of individual subsystems are logically composed to build more complex procedures upto a full mission design. While the system's performance can be checked using realistic simulations, crucial properties such as dead-lock avoidance, safety and liveness are formally verified at both levels, using in particular some advantages of synchronous programming and associated tools. The approach is illustrated through simulation results while experiments are in progress.

1 Introduction

Underwater manipulation and inspection tasks are commonly encountered in maintenance or decommisioning of underwater structures, e.g. in oil or subsea cables industries. Although some applications in shallow water can be carried out by divers, underwater robots have increasingly been used since the late 1970s as cost effective systems in both deep and shallow water.

However currently used Remotely Operated Vehicles exhibit a very low level of automation thus increasing the load of the operator and the overall time of operation. For example, before an inspection task with a teleoperated arm the R.O.V. must be clamped on the underwater structure with purpose built sucking pads after a cleaning operation with a water jet.

*This work is funded by the Esprit III Basic Research Action UNION

It is expected that increasing the level of automation can increase the efficiency of underwater manipulation systems by avoiding the design of specialized devices and by cancelling unuseful tasks. In the framework of the CEC UNION Basic Research Action ([9]) preliminary research was conducted to assess active stabilization of the underwater platform using exteroceptive sensors and closed-loop control, thus building virtual links between the vehicle and the environment rather than mechanical ones. The experimental plant under consideration is made of the Ifremer's Vortex vehicle fitted with a 7 d.o.f. arm while the control structures were designed following the ORCCAD approach. We present here a preliminary study to evaluate the possibility of stabilization of a free-floating vehicle using sensor-based control during arm motions.

This paper is organized as follow: in the next section we recall the main requirements of a robotic control system and the corresponding features in ORCCAD. In section 3 the experimental plant and planned experiments are described. Section 4 details the design process of basic actions of the system. These basic actions are further composed to build more complex procedures as explained in section 5. Guidelines for the future are given in the conclusion.

2 The ORCCAD Architecture

Requirements A complete robotic system includes a lot of various subsystems coming from various fields of science and technology like automatic control, sensor data processing and computer science. The goal of the control architecture is to organize 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 pure

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 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.

Therefore robotic systems belongs to the class of hybrid reactive and real-time systems which need to use a lot of different methods and tools to be programmed and controlled. The ORCCAD[12] environment is aimed to provide users with a set of coherent structures and tools to develop, validate and encode robotic applications in this framework.

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 characterizes in a structured way continuous time closed loop control laws, along with their temporal features related to implementation and the management of associated events. These events are pre-conditions, post-conditions and exceptions which are themselves classified in type 1 (weak), type 2 (strong) and type 3 (fatal) exceptions.

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. The characterization of the interface of a RT with its environment in a clear way, using typed input/output events, allows to compose them in an easy way in order to construct more complex actions, the *Robot-Procedures* (RPs): briefly speaking, they specify 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.

The *Robot-Procedure* (RP) paradigm is used to logically and hierarchically compose RTs and RPs in structures of increasing complexity. Usually basic RPs are designed to fulfill a basic goal through several potential solutions e.g a mobile robot can follow a wall using predefined motion planning or visual servoing or acoustic servoing according to sensory data availability and analysis. RPs design is recursive 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 allows to systematize and thus automatize 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 [4].

3 Experimental plant and scenario

Let us now focus to the significantly complex case study which will be considered in the sequel. This example is currently developed in the framework of the Union project[9] to assess the design and verification methods of Orccad with an underwater testbed application. The mission will take place in a pool, using the Vortex vehicle fitted with a manipulator.

Vortex is a Remotely Operated Vehicle (ROV) designed by Ifremer as a testbed for control laws and control architectures. It is equipped with a set of six screw propellers and with a traditional sensing set such as compass, inclinometers, depthmeter and gyrometers allowing to measure most of its internal state. A video camera is used for target tracking tasks and a belt with eight ultrasonic sounders allows to perform positioning and wall following tasks. Vortex is also equipped with an electric powered Mitsubischi Pa10 arm with 7 degrees of freedom to perform manipulation tasks.

The vehicle control algorithms are computed on an external VME backplane running Pirat, a C++ library of objects dedicated to underwater vehicles con-

trol. At a higher level, i.e. control laws and mission management, the reactive synchronous ESTEREL language ([2] is used to design Robot-tasks and Procedures, consistently with the ORCCAD approach. The control algorithms for the arm are run on a second VME backplane. As the two controllers only have a low bandwidth communication capability, control laws for Vortex and the Pa10 arm run independently and only short synchronization messages are exchanged on the communication link. This system depicted in figure 1 is used to test the ORCCAD approach given the following informal end-user's requirements.

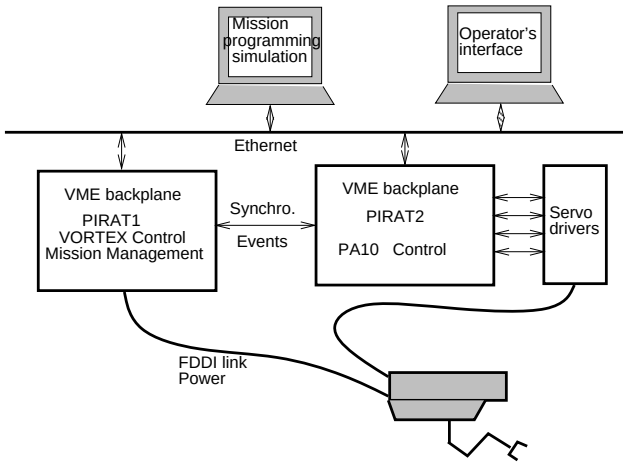

Figure 1: An underwater experimental plant

Starting from the initial position, Vortex is set in station keeping mode. After completion of all necessary initialization it navigates in wall following mode up-to reaching the working area. The working phase is then started to inspect some locations on a pipe with the arm tip while Vortex is stabilized in a pool's corner facing the pipe to inspect. Once the arm operation is finished an operator call sets it again in parking position while the vehicle comes back to its initial position under the crank.

At any time the detection of a water leak or of an hardware failure must lead to a mission abortion leaving the system in a safe situation, i.e. setting an alarm and emergency surfacing with the arm locked in folded position. Other exceptions more specific to a given system or control algorithms are also defined inside the subtasks involved in the mission.

This mission scenario naturally lead to split the mission in five main procedures: INITCRUISECONFIG consists in preparing the vehicle for cruising. REACHWORKINGAREA is used to navigate in the pool until the vehicle reaches the inspection place. DOINSPECTION is used to coordinate actions of the platform and of the arm in order to accomplish the main task which is inspecting the pipe. GOHOME is in charge of driving the vehicle to its homing position and preparing it to be pulled out. EMERGENCY is always active to handle permanent recovery behaviors like triggering a fast ascent in case of water leak. Figure 2 summarizes the tasks and procedures which were designed to achieve this mission.

Procedures	Sub−Proc. and Tasks	Main Events	Comments
InitCruiseConfig	Vortex: StationKeeping Pa10: GoToPark	Parked	Using absolute position Arm folded and locked
ReachWorkingArea	Vortex: SwimAhead WallFollowing Pa10: GoToPark	WallDetected CornerDetected	Navigation Using US sensors
DoInspection	Vortex: KeepStableBase Pa10: Pa10_Working Pa10BrakesOff	StopKeepStable Operator's calls	Stabilization in workspace Inspection with arm tip Brakes are released
GoHome	Vortex: GoToPoint Pa10: GoToPark	WayPointReached	Come back to initial position
Emergency	Vortex: GoUp Pa10: GotoPark	Hardware failure Water leak Operator's interrupt	Emergency recovery behaviour
KeepStableBase	Vortex:KeepStableCamera KeepStableUS	TargetFound TargetLost	Visual servoing Distance towalls servoing
Pa10_Working	Pa10: Pa10_TT_JS Pa10_TT_SE3 Pa10_Teleop	JointLimits Operator's calls	Traj. tracking in joint space Traj. track. in operationnal space Teleoperation mode
GoToPark	Pa10: Pa10_TT_JS Pa10BrakesOn	ParkPosReached ArmLocked	Arm folded Arm locked by brakes

Figure 2: Summary of tasks and procedures

4 Control laws and Robot-tasks design

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 Robot-task library. Ideally, from the control point of view the Vortex/PA10 system should be considered as a single global 13 d.o.f. dynamic system. However, due to the lack of a suitable and well calibrated model of the global backward dynamics we choose in this preliminary study to separately control the vehicle and the arm having hope that each control law will be robust enough to remain stable despite the disturbances due to the inertial coupling. Let us now describe some of the control laws and RTs which were designed for the PA10 arm and the Vortex R.O.V., both being based upon the Task-function approach [11].

Modular Specification in Continuous Time The control law which is used for the PA10 manipulator belongs to the class of decoupling/feedback linearization in a dedicated task space. Its general expression is given in (3). The goal assigned to the ma-

nipulator is defined as the regulation to zero of an $n-$dimensional output function $e(q,t)$, where q is a vector of generalized coordinates, aimed to represent in a clever way the user's objective.

In the present case, the output function includes the tracking by the arm tip of a trajectory in the 6-dimensional space of frames (SE(3)). Since the robot has 7 joints, one degree of freedom is available for achieving simultaneously a secondary task, which can be expressed as the minimization of a scalar cost $h_s(q)$. Classical secondary goals of that kind are the avoidance of kinematic singularities or of joint limits, the minimization of the velocity norm, etc.... The two tasks are finally gathered into a single one through the expression:

$$e = J_1^{\dagger} e_1 + \alpha (I_6 - J_1^{\dagger} J_1) \frac{\partial h_s}{\partial q} \tag{1}$$

where e_1 expresses the trajectory tracking task:

$$e_1 = \begin{pmatrix} P(q) - P^*(t) \\ A(q,t) \end{pmatrix} \tag{2}$$

A being a parameterization of the attitude error, P the position of the tip and α a positive weighting factor. $J_1 = \frac{\partial e_1}{\partial q}$ is the jacobian matrix of e_1.

The final control law is given below.

$$\Gamma = -k\hat{M}\left(\frac{\widehat{\partial e}}{\partial q}\right)^{-1} G\left(\mu De + \frac{\widehat{\partial e}}{\partial q}\dot{q} + \frac{\widehat{\partial e}}{\partial t}\right) + \hat{N} - \hat{M}\left(\frac{\widehat{\partial e}}{\partial q}\right)^{-1} \hat{f} \tag{3}$$

Here, Γ is the array of control actuator torques, M and N gather the Lagrangian dynamics, and k, μ, G, D are tuning parameters. The "hats" indicate that more or less complex models of the concerned terms can be used. In fact, it should be emphasized 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 ([12]) available through the graphical interface depicted by figure 3. On the basis of these information and requirements, a set of objects defined in the RT modeling[5] can now be instantiated by setting adequate numerical values. A complete description of a similar RT may be found in [7].

The task-function approach can also be used to specify sensor-based tasks, i.e. where the d.o.f. of the robot are regulated with respect to the environment. Too such control laws have been developed for Vortex: stabilization in a pool's corner using acoustic sensors and visual servoing facing the pipe. They are detailed in [15] and [10] respectively.

Time-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, duration of computations, communications and synchronizations between the processes. It is also finally necessary to enter the localization of data processing codes.

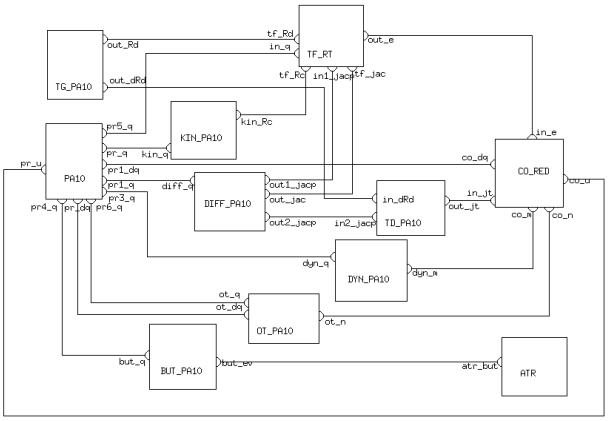

Figure 3: Graphical Representation of the Robot-task Specifying a Trajectory Tracking Action in the Frame Space

At this stage of specification the basic entities are Module-tasks, i.e. real-time tasks, usually periodic, which implement an elemental functional module of the RT. Since MTs may, possibly, be distributed over a multi-processor target architecture they communicate using typed ports and specific synchronization mechanisms. As inter-tasks synchronization may lead to dead-locks a Petri net model of the synchronization skeleton of the RT can be automatically build. Its analysis allows to conclude with dead-lock freedom and with the temporal coherency of the design [14].

The logical behaviour of the RT is automatically encoded through a graphical window. Thanks to the strong typing of events and exceptions the code generator was proven to be correct and thus guarantee that crucial properties like safety and liveness are true. ([5]).

Simulation of a Robot-task Simulations can take place after this first logical verification phase. We now present a few simulation results concerning the global underwater system.

Compared with others simulation softwares commonly used in robot control a main characteristics of

Simparc ([1]) is that it allows to simulate both the plant dynamics and important temporal characteristics of the controller like sampling rates and communication delays, including the basic features of real-time operating systems. Figure 4 shows the model of the controller of the underwater system depicted in figure 1. The coupled dynamics (including drag, lift and hydrostatic forces) is computed using some classes of the Dynamechs software ([8]) integrated in the former model of Vortex. The simulation uses ray tracing models for the video and acoustic sensors. According to the limits of the actual hardware the sampling rates of the vehicle control and arm control are set to 100ms and 10ms while the acoustic sounders are triggered every 360ms.

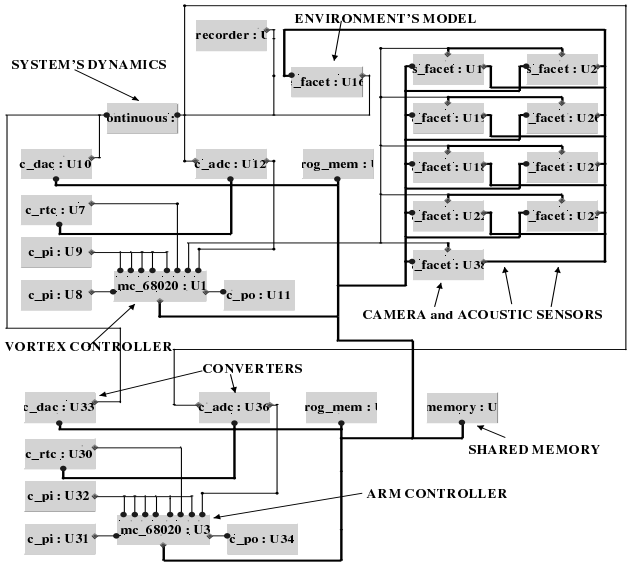

Figure 4: Simulation model of the underwater system hardware

Figure 5 shows the pitch and roll orientation errors of the vehicle stabilized in a pool corner using the sounders during a fast motion of the arm. The first and second joints are turned by 90 degrees in 3 seconds. Thanks to a rather large hydrostatic stability of Vortex, the vehicle does not capsize even if it is not actively controlled. However, closed-loop stabilization with the U.S. sensors reduces both transient and steady-state errors.

Tuning some parameters of the simulation file rapidly shown that the main limitation in the ROV stabilization performance come from the low sampling rate of the acoustic sensors. Therefore improving the stability of the underwater platform would better come from an upgrade of the hardwired electronics of the sensors rather than first increasing the computing power, a very interesting information for the system designer.

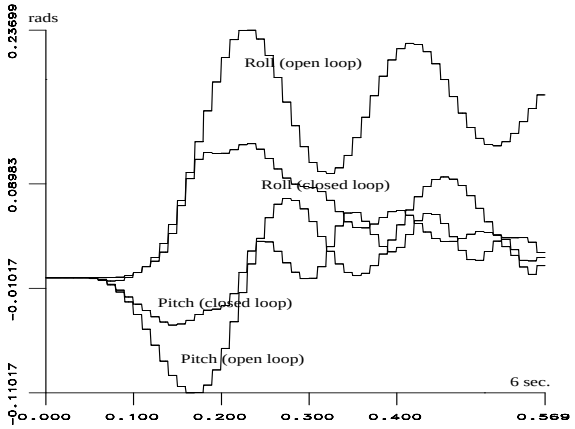

Figure 5: Roll and pitch errors during arm motion

5 Design and Analysis of Procedures

We now detail the specific procedure DoInspection in order to enlighten the specification and analysis process we propose for basic actions composition.

The DoInspection procedure aims at performing manipulation operations with the arm while ensuring the stability of the base. Since the base and the arm are controlled separately despite of mutual disturbances, an important requirement is that the arm can only move when the base is stable enough and stops its motion otherwise. Motions of the arm can be performed again after recovering platform stability. It is clear that the need to handle concurrency, preemption and synchronization is very strong in this procedure as well as the stability of the overall physical system; these reasons justify our choice to present it in detail.

Since the camera sampling rate is about three times faster than the U.S. sensors, it is expected that the first one will be more robust with respect to the disturbances coming from the arm motions. However, if the camera looses the target, we can switch to sounders stabilization until being able to recover the target 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.

More precisely, the procedure can begin when the vehicle has reached the working area, i.e. it is more or less accurately positioned in front of the cylinder to be inspected. During arm operations the vehicle is stabilized using the KeepStableCamera task (a

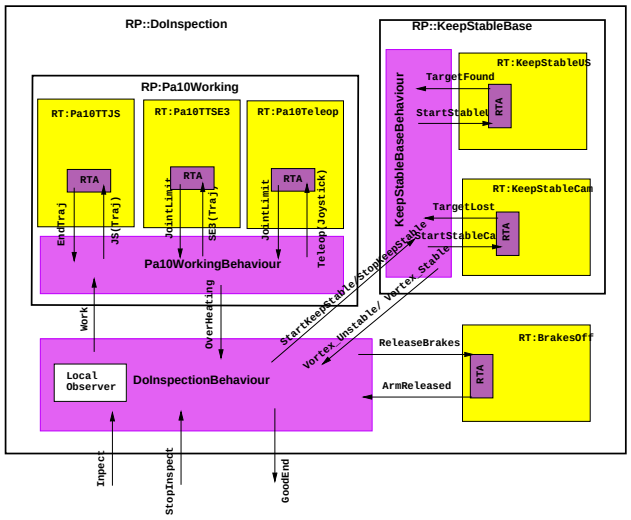

Figure 6: Outline of the DoInspection procedure

vision-based servoing task using geometric features of the cylinder). The presence of the video signal is monitored all along the procedure. Loosing this signal, e.g. due to an uncontrolled motion of the vehicle, raises a *TargetLost* signal switching to the KeepStableUs task up to stability and video signal recovery (*TargetFound*).

Once the vehicle is stable in front of its target the actions concerning the arm can be run concurrently with the platform stabilization on the second controller. The Pa10Teleop may begin under control of predefined signals coming from the operator's interface and keyboard to touch some locations on the target and store their coordinates. Upon an operator's command, the arm controller is set in the mode of automatic trajectory tracking in operational space (Pa10ttse3 task) to point again the visited locations. In both modes, a platform unstability temporary blocks the arm motions (Pa10BrakesOn task) up to recovering vehicle control. The corresponding control code is given down below. It is written using a mission programming language currently designed in the UNION project [16].

```
PAR{
SEQ(do
      Pa10BrakesOn
   until  BaseStabilized;
   repeat 2 times
      Pa10SafeMoveJS;
      Pa10SafeMoveSe3;
   end repeat
   ;
   do
      Pa10SafeMoveJs
   until  ParkingPositionReached
   emit  InpectionOK
   ) ;
```

```
SEQ(do
      loop
         BaseSearchTarget;
           emitBaseStabilized
         KeepStableBase;
      end loop
      until  (InpectionOK)
   ) }
```

The left branch of the parallel statement states that the Pa10 actuators are blocked until the base is stable. A soon as this condition is satisfied the arm is driven to inspect two locations in two steps: a motion in the joint space drives the arm in the region of interest and sequently a new motion in the tip frame space moves the end-effector in front of the inspection point. We remark that the meaning of 'safe' motion is that the base and the arm do not move concurrently. Finally, the arm reaches the parking position and signals that the inspection is finished.

The second parallel branch states that the base is situated in front of the target such that the axis of the cylinder is exactly in the center of the camera's image with the right appearance. Sequently, it signals that the base is in a motionless position and keeps it using KeepStableBase. We remind that this procedure handles the loss of the video signal and that the 'loop' instruction aims precisely to come back to the vision based control. *InspectionOK* signal, emitted by the branch of the parallel instruction handling the Pa10 activity interrupts the stabilization of the base and the procedure terminates.

At compile time the control code of this procedure expands in more than 300 Esterel statements thus alleviating the burden of the programmer while preserving formal verification capabilities.

Logical Behaviour Verification Firstly, the satisfaction of *crucial properties* is checked. Concerning the *safety property* (any fatal exception must always be correctly handled) we proceed as follows. Knowing the user's specification defining the fatal exceptions and the associated processing, we build a criterion defined by abstract actions like:
"$Error = /Water\_Leak?\ and\ not\ /Accent!$".
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 bisimulation to a one state automaton with a single action, the "Success" one.

Conflicts detection We are interesting here to check that during the RP evolution it does not exist instants where two different RTs are competing for using a same resource of the system. We consider the physical resources controlled by the RTs (the arm and the vehicle) as well as the software resources used by the

controllers (real-time tasks). For example, we want to verify that the RTs KeepStableCamera and KeepStableUs never compete to apply different desired force screws to Vortex thrusters during all the RP evolution. We reduce the global automaton to the only interesting signals *StartServoingVortex* and *StoppedServoingVortex*. Thus we 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.

Finally, the *conformity of the RP behaviour with respect to the requirements* is verified. We want to certify that the arm is motionless when the platform is recovering stabilization using the ultra-sonic sensors. The actions involved in this property are KeepStableUs and Pa10ttse3 and therefore the signals in respect of which we observe and reduce the global automaton are *STARTKeepStableUs*, *BFKeepStableUs*, *STARTPa10ttse3*, *BFPa10ttse3* and *AbortPa10ttse3*. The resulting abstract view is given figure 7. We remark that the two tasks always run in sequence: from the state 'oc3' either KeepStableUs is executed (state 'oc4') or Pa10ttse3 (state 'oc5'). In particular, if KeepStableUs run from state 'oc5' the Pa10ttse3 task is aborted and only re-activated after the end of the KeepStableUs task.

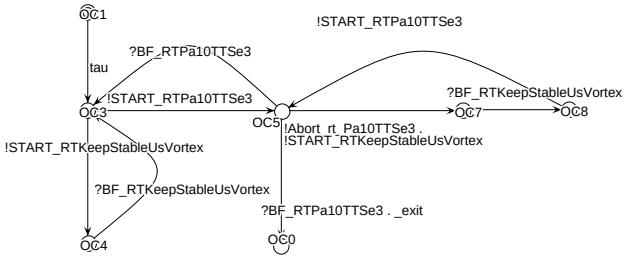

Figure 7: Abstract view at RT level of DoInspection

Validation of Smooth Switching After the verification of the logical behaviour of the procedure we focus now on a particular phase of the system's evolution: the transition between two control tasks. The switching mechanism must ensure that two control laws are mutually exclusive, i.e. they never compete to control a given subsystem, and that the transition must be as fast as possible to maintain the system's stability ([13]). This mechanism manages the real-time tasks involved in the successive RTs and is also encoded using Esterel (only for single processor/single rate tasks up to now). The abstract view given figure 8 shows that the stopping and starting signals are correctly ordered and that the transition takes place in a single automaton transition thus minimizing the latency.

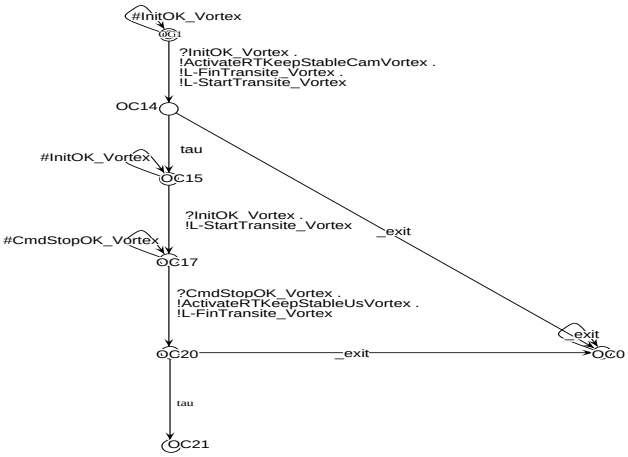

Figure 8: Abstract view of the switching mechanism

Logical analysis provides no information about the transient behaviour of the physical system during RT switching. Structures to generate simulation programs of RP as sequences of RTs were also developed. A characteristic of these simulations is that the simulated discrete event controller is the same as the one which will be used for execution in real world[6].

Figure 9 presents a 15 sec simulation of the DoInspection procedure. The two plots at the top of the figure illustrate the evolution of the x and y coordinates of the base, while the lowest plot represents the evolution of the x coordinate of the origin of the end-effector frame of the arm. Vertical lines indicate the instants of detection of the aforementioned events which imply a Robot-task switching in the procedure. The results appear to be satisfactory since the switching is smooth enough to allow the video signal recovery.

6 Concluding remarks

Although the mission we describe in this paper looks simple compared with a real operation at sea, the interleaving of events and activities of the full mission automaton justifies our hierarchical design and verification process, where we design complex actions from already validated ones lying at a lower level. As an illustration of the complexity, let us mention that the overall automaton of the application has 715 states and 8,597 transitions, which clearly excludes to build it by hand or to verify it visually. This size is a consequence of the existence of a true parallelism between arm and vehicle controls all along the operation. It

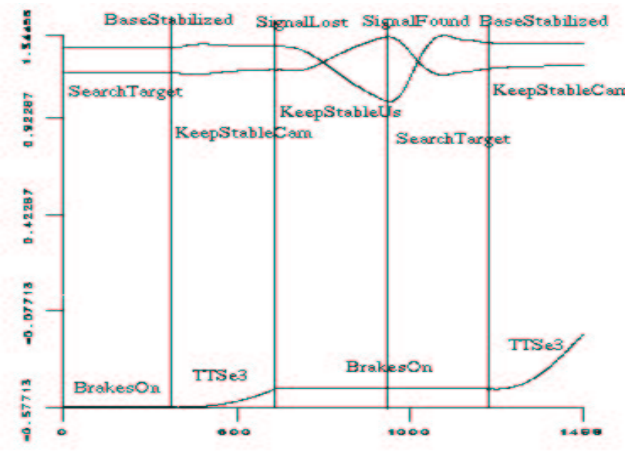

Figure 9: Simulation of DoInspection

is expected that distributing the control code as synchronized automata[3] could help to reduce this complexity while preserving the formal verification capability. This solution is currently implemented on the Vortex/PA10 controller.

Since tools for the verification of temporal features and hybrid systems are not mature enough, we only verify the logical behaviours of robotic actions involved in the application. Thus, realistic simulations remains useful to check numerical properties, i.e. tasks synchronization delays or control law performance. Previous experience has shown that this kind of simulation is helpful to prepare experiments and to save time on the real site. Experiments are currently being carried out with the real system in Ifremer's pool and it is expected that preliminary experimental results will be available by the time of the conference.

References

[1] C. Astraudo, J.J. Borrelly, "Simulation of Multiprocessor Robot Controllers", *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, May 1992.

[2] G. Berry, G. Gonthier: " The Synchronous Programming Language ESTEREL: Design, Semantics, Implementation", *Science Of Computer Programming*, Vol 19 no 2, pp 87-152, 1992.

[3] G. Berry: "Communicating Reactive Processes", in *Proc. 20th ACM Conf. on Principles of Programming Languages*, Charleston, Virginia, 1993.

[4] B. Espiau, K. Kapellos, M. Jourdan, and D. Simon *On the Validation of Robotics Control Systems. Part I: High level Specification and Formal verification*, Inria Research Report no 2719, November 1995

[5] K . Kapellos: *Environnement de programmation des applications robotiques réactives*, PhD dissertation, Ecole des Mines de Paris, Sophia Antipolis, France, November 1994.

[6] K. Kapellos, S. Abdou, M. Jourdan, B. Espiau "Specification, Formal Verification and Implementation of Tasks ans Missions for an Autonomous Vehicle" *4th Int. Symp. on Experimental Robotics*, Stanford, USA, June 30- July 2, 1995.

[7] K. Kapellos and B. Espiau, *Implementation with Orccad of a Method for Smooth Singularity Crossing in a 6-DOF Manipulator*, Inria Research Report no 2654, September 1995

[8] S. McMillan, D. Orin and B. McGhee, "Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 25, no 8, August 1995.

[9] V. Rigaud e.a., "UNderwater Intelligent Operation and Navigation: Main objectives and first results", in Proc. *6th IARP workshop on Underwater Robotics*, Toulon, France, March 1996.

[10] P. Rives and J.J. Borrelly: "Visual Servoing Techniques Applied to an Underwater Vehicle", submitted to Int. Conf. on Robotics and Automation, 1997

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

[12] D. Simon, B. Espiau, E. Castillo, K. Kapellos: " Computer-aided Design of a Generic Robot Controller Handling Reactivity and Real-time Control Issues", *IEEE Trans. on Control Systems Technology*, vol 1, no 4, December 1993.

[13] D. Simon, K. Kapellos and A. Santos: "Smooth task switching mechanisms", UNION project deliverable no D2.5, September 1995.

[14] D. Simon, E. Castillo and P. Freedman, "On the Validation of Robotics Control Systems. Part II: Analysis of real-time closed-loop control tasks", Inria Research Report no 2720, November 1995

[15] D. Simon, K. Kapellos and B. Espiau: "Control Laws, Tasks and Procedures with ORCCAD: Application to the Control of an Underwater Arm", in Proc. *6th IARP workshop on Underwater Robotics*, March 1996, Toulon, France.

[16] N. Turro and E. Coste-Manière: "Mission Programming Language", UNION project deliverable no D2.2, July 1996.