A Control Architecture for Contextual Tasks Management: Application to the AUV Taipan.

A. El Jalaoui, D. Andreu, B. Jouvencel
LIRMM - Montepllier II University
161, rue Ada, 34 392 Montpllier Cedex 5 (France)
Email: (eljalaoui, andreu, jouvencel)@lirmm.fr

Abstract— **A new method for control architectures design of underwater robots is presented. The approach proposed tries to meet aims related to the design of software controller for embedded real time system namely: modularity, evolutionarity and robustness. This control architecture, applied to the AUV TAIPAN, will be presented.**

I. Introduction

The need to operate in deep water leads research to concentrate on an increasing level of autonomy, in order to complete a mission without the assistance of the human operator.

On a technological point of view AUV (Autonomous Underwater Vehicle) are based on a set of embedded computer resources and a set of sensors/actuators that can change according to the mission to be performed (for example pipeline inspection, cartography, bathymetry...). The robot should be comparable to a general-purpose vehicle adaptable to different tasks, and consequently evolving in accordance with the technological progress or the appearance of new challenging scientific applications.

Control software developed for this kind of vehicle becomes complex and requires a methodology of design. Control architectures are usually classified into three main categories: deliberative, reactive and hybrid [12]. Deliberative architectures are based on planning using an environment model. Reactive architectures are based on a set of active behaviours. Hybrid architectures are a mixed of two previous.

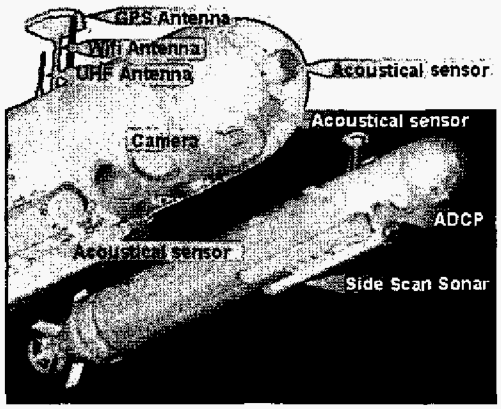


Fig. 1. Taipan II

Three main criteria are generally used to evaluate an architecture:

- **Modularity:** a complex software should be divided into components which can be individually designed, implemented and tested.
- **Evolutionnarity:** when changing the application or embedded devices (sensors for instance), the architecture should be upgradeable.
- **Robustness:** the vehicle must be able to reliably execute planned sequences under conditions of uncertainty, to rapidly respond to unexpected events.

This study aims at introducing a new methodology of control architecture design and its application to AUV TAIPAN shown at figure 1. Taipan is an AUV developed at LIRMM, France. It is a small size and low cost torpedo-shape AUV, used in very shallow water applications. Its dimensions are 1.9 m of length, a diameter of 0.25 m and a weight of 40 kg (for more details see [13]).

This mixt architecture consists of three layers (fig. 2): *global supervisory control*, *local supervisory control* (one for each mode: autonomous, teleoperation, cooperation) and *low level control.*

Each level handles objects corresponding to his level of abstraction, three types of object are used (fig. 3): the global supervisor receives a set of *objectives* (defining the mission) from the operator and it transmits a set of *sub-objectives* to the dedicated local supervisor . The local supervisor gives to the low level control and instrumentation modules a set of *orders.*

The first section details the three layers of this software control architecture and objects they handle. The second section deals with the way in which a given mission is performed by taking account of the context.

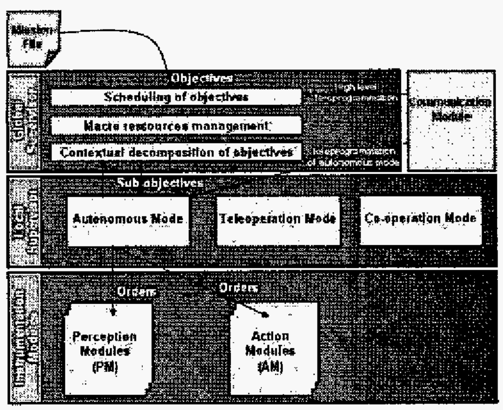


Fig. 2. Mixt control architecture

II. Control Architecture

A. Description

1) Global supervisor: The Global Supervisor (GS) receives on behalf of the user a file containing

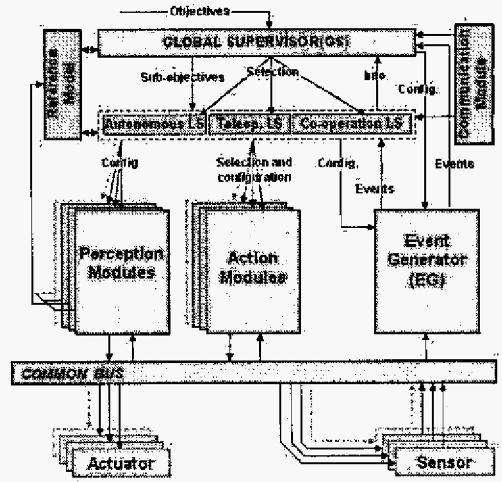


Fig. 3. Software control architecture

a mission to perform. A mission is a succession of objectives that the system must achieve during its course. Objectives can be a motion or others activities to be carried out at a given time (bathymetry for example). These objectives can be carried out successively or in parallel according to their nature. Information relating to their scheduling is specified by the user before the beginning of the mission.

The way to build an objective is described section II-C.1. This technique allows the GS to schedule correctly all the objectives.

The GS ensures that objectives, to be simultaneously performed, do not use the same resource. For example the GS prevents the objectives *go to surface* and *inspect pipeline* to be performed in same time because they use the same set of resources {*propeller, body, fins*}. More precisely this set of resources is defined as a macro-resource.

On the other hand, the objectives *inspection pipeline* and *cartography* respectively use the macro-resource {*propeller, body, fins*} and the resource *Side Scan Sonar.* Thus they can be launched simultaneously. Resource management ensures that contradictory orders won't be sent to actuators (by means of control modules).

During the achievement of an objective, the AUV crosses several steps. These steps achievement can rely on different control laws as for instance to *dive*, to *inspect pipeline*, ... So we have to decompose each objective in more detailed description called *sub-objectives*. For example the objective *inspect pipeline* is broken up into the sequence {*dive, go to, search for pipeline, follow pipeline*}.

Finally the GS sends to the concerned local supervisor (LS) the sub-objective; the LS will return an execution report.

2) Local supervisor: A local Supervisor (LS) is dedicated to the control of a resource in a given mode. As far as TAIPAN is concerned, we have only one resouce (the vehicle) which has three different operating modes:

- *autonomous mode:* it performs sub-objectives from GS,
- *teleoperation mode:* low level teleoperation from an operator (used only at this time, when the AUV is in surface)
- *cooperation mode:* it controls the AUV in a flotilla.

This paper deals only with the autonomous mode.

As previously stated in section II-A.1, the LS receives sub-objectives to be performed (e.g. *dive, go to*). In order to achieve a sub-objective we need to collect data from sensors and to send commands to actuators. For each sub-objective execution, the LS uses control and instrumentation modules.

3) Low-level control modules: There are two kinds of organs in an AUV, sensors and actuators. The first are managed by Perception Modules (PM) and the second by Action Modules (AM). All these components as well as the event generator EG (cf. fig.3) and the instrumentation use a common bus to exchange data.

A PM is built for each data type (called variables) which is required in the architecture (e.g. position x,y,z,ϕ, θ, ψ). It is often necessary to use data proceeding from several sensors measures to calculate or estimate precisely a variable. Thus, a PM can manage several sensors.

Two kind of jobs are achieved by a PM; the first concerns sensors configuration (starting, switching off, mode change), the second concerns the data processing.

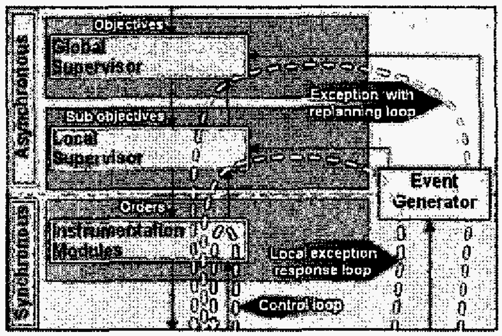


Fig. 4. Reaction loops

AM, which contains in most case control laws, allows to compute commands to send to actuators.

All these modules are activated and configured by the LS. Indeed to activate a given perception, the LS sends an order to the corresponding PM and to apply a control on an actuator it sends an order to the corresponding AM. PM and AM then run in a periodical way.

Perception variables (estimated or measured) are exchanged between the low-level modules (AM, PM and EG) by means of the common bus.

In section II-C.3, we will deal with problems of interference occurring when using interfering adjacent sensors (as ultra-sound based sensors).

B. Reaction loops

As depicted on figure 4, there are several *reaction loops* within this architecture.

1) Control loop: The AUV is a system which continuously interacts with its surrounding environment while trying to achieve a goal. The first relevant reaction loop is control loop. By means of perception modules, the system gets back data, which inform about the AUV situation. Then according to the state it has to reach (given by the LS), it acts through AM on actuators.

2) Local exception response loop: Certain events reflect a dangerous situation and need a fast response. These events are built by the *Event Generator (EG)* from PM variables.

This component, the EG located between the LS and the Instrumentation Modules (then between the asynchronous and synchronous parts of the architecture), monitors variables produced by the PM in order to detect relevant events for the LS.

The EG can be configured by others control architecture components to inform them of the occurrence of certain events.

In order to provide a fast response, these events are treated by the LS. For example, during the navigation stage, if the AUV meets an obstacle, the LS suspends the current job, launches the process which allows to avoid obstacle.

3) Exception with replanning loop: Another events cannot be carried out by the LS as for example a water leakage or a propeller failure. In these cases, the event is treated in GS level. According to the kind of event, the GS can change the current objective or stop the current mission.

C. Objects used in the architecture

1) Objective: As stated in section II-A.1 a mission is described by the operator as a set of objectives, which are then arranged to constitute a sequence. This sequence is then executed by the GS. Information related to the way in which the controller must arrange the objectives to achieve the mission are contained in the objects *objectives* themselves. An objective can be launched after a fixed time (potentially null) after two kinds of events:

- **Particular state of the system** (e.g. Torpedo in water),
- **End of another objective**

Moreover, an objective can stop itself or (after a fixed time) after the end of another objective.

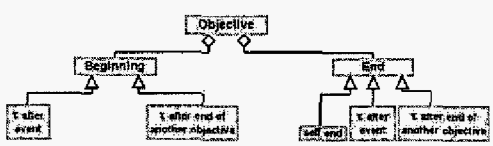


Fig. 5. The object *Objective*

2) Sub-objective: The sub-objectives, obtained by decomposition of objectives, contain information required to the execution of actions. They contain information relating to the use of the instrumentation modules: list of PM and AM to be used and their configuration parameters.

3) Orders: The orders are produced by the SL for the AM and PM. When a PM receives an order, he has to make a data acquisition according to parameters contained in this order, which can be the sensor mode or the type of data processing (filtering, ...). An order received by an AM contains the control law parameters.

III. Contextual Task Management

A. Mission described by a set of objective

As described in section II-C.3, a mission is made up of a set of objectives that the AUV must achieve. The mission achievement consists with the execution of this logico-temporal sequence of objectives.

Example of mission:

- **Wait** 30s
- **Go to** A($x_A; y_A; z_A$)
- **GPS based position updating**
- **inspect pipeline** at B + **salinity statement**
- **bathymetry** at C
- **End**

The description of this mission by the user requires the implementation of five objects *objective* to wit: *Navigation*, *GPS updating*, *Inpect pipeline*, *Salinity* and *Bathymetry* (fig. 7).

To achieve this mission, the AUV will have to behave as shown at fig. 6

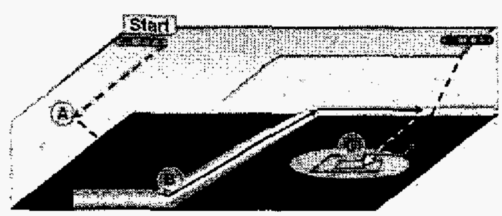


Fig. 6. Example of mission

During the stage of pipeline inspection, the pipeline can be occulted. The GS reaction in this case could be to stop the mission and to proceed to an emergency surface. However, losing pipeline position is not a dangerous situation for the system,

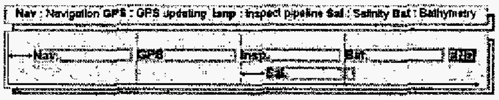


Fig. 7. Objectives scheduled by the global supervisor

it could finish the mission by carrying out the remaining objectives.

To make it possible, we add to the object objective (described in section II-C.1) another event which can launch it, to wit the failure of another objective (fig. 8). Thus the user has to specify, in the mission file, two objects *batymetry*. The first will be launched if the objective *inspect pipeline* ends normally and the second will be launched if it fails.

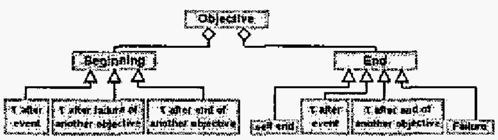


Fig. 8. The object *Objective*

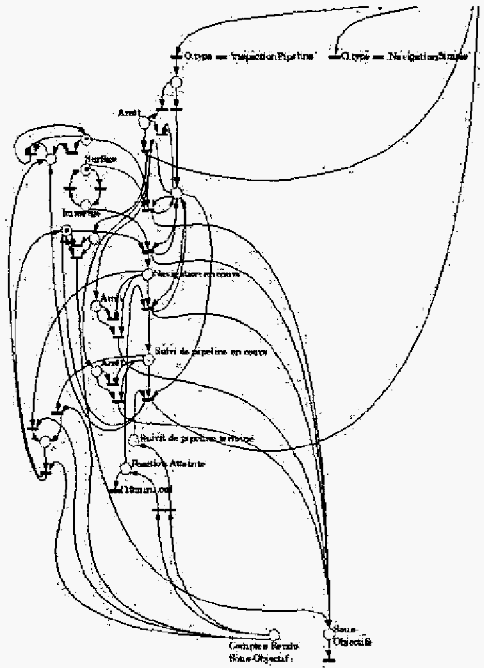


Fig. 9. Partial Petri Net for decomposition of objectives

B. From the objective to the sub-objectives

When an objective is selected to be executed, it is decomposed into a sequence of sub-objectives. For example as shown in figure 10(a) the objective *inspect pipeline* is broken up into the sequence : {*dive, go to, search for pipeline, follow pipeline*}

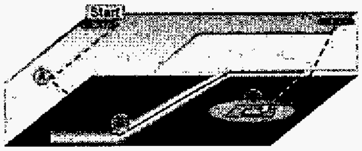
However, this decomposition should be adapted to the context. As shown in the figure 10(b) the torpedo is already in the water and in order to achieve the objective *inspect pipeline*. In this case, the decomposition must be {*go to, search for pipeline, follow pipeline*}. The sub-objective *dive* has been removed. We are therefore led to make a context-dependent decomposition. This functionality is ensured in the global supervisor by the object Petri net shown at the figure 9. Each sub-objective allows, either to cross the system from one state to another (e.g. state *topedo in surface* towards *torpedo in immersed*), or achieve a precise task (e.g. salinity statement). The Petri Net shown fig.9 takes into account the current system state as well as the sub-objective achievement to generate the next sub-objective of the sequence quoted.

C. Management of instrumentation

At the local supervisor level, each sub-objective generates one or more order periodically. For example the sub-objective *dive* produces two orders, one for the PM who gives current depth value, the other for the AM managing the propeller and fins.

The main problem arises using several sensors that are interfering (for instance at fig. 1 the Side Scan Sonar and another acoustical device can both interact).

In fact, we generally do not need to continuously have the value of a variable given by a PM. Of course when this value is required by a given control law (within an AM), it is periodically needed, according to a period imposed by this control. It is then necessary to avoid executing, in a same period,



(a) Surface

(b) Torpedo in water

Fig. 10. Contextual decomposition of objectives

PMs using interfering sensors.

The LS is therefore led to schedule the activity of PMs, in order to avoid the simultaneous execution of two PMs that are using interfering sensors.

IV. Conclusion

The presented work belongs to the field of software control architecture. A control architecture, particularly developed for Autonomous Underwater Vehicle has been proposed. Modularity, evolutionnarity and robustness are main criteria of this mixt architecture design.

A simple language (composed by a set of objects called *objectives*) is supplied to the user for building a mission. This set of objectives is built thanks to sub-objectives. The sub-objectives reflect the capabilities of the given AUV. The change of AUV action or perception capabilities involves potentially the appearance of new objectives and thus makes the language upgradeable.

References

[1] J.D. Carbou, D. Andreu, P. Fraisse, "Events as a key of an autonomous robot controller" in *15th IFAC World Congress (IFAC'b 02), Barcelone, Spain, 21-26*, july 2002.

[2] A. Healey, D.B. MARCO and R. McGHEE, "Autonomous underwater vehicles: Hybrid control of mission and motion," in *Autonomous Robots 3, 169-186*, 1996.

[3] P. Ridao, J. Yuh, J. Batlle, K. Sugihara, "On auv control architecture," in *Proceedings of the International Conference on Robots and Systems*, 2000.

[4] A. J. Healey, D. B. Marco, P. Oliveira, A. Pascoal, V. Silva, and C. Silvestre, "Strategic level mission control - an evaluation of coral and prolog implementations for mission control specifications," in *Proceedings of the 1996 Symposium on Autonomous Underxater Vehicle Technology, pp.125-132, Monterey California*, 1996.

[5] P. Oliveira, A. Pascoal, V. Silva, C. Silvestre, "On the design and development of mission control systems for autonomous underwater vehicles: An application to the marius auv," in *The Sixth International Symposium on Robotics and Automation*, 1996.

[6] P. Oliveira, A. Pascoal, V. Silva, C. Silvestre, "Design, development, and testing at sea of the mission control system for the marius autonomous underwater vehicle," in *The Sixth International Advanced Robotics Program IARP-96*, 1996.

[7] P. Oliveira, A. Pascoal, V. Silva, C. Silvestre, "Mission control of the marius auv: System design, implementation, and sea trials," in *International Journal of System Sciences, pp.1065-1080*, 1998.

[8] J. L. Claude Barrouil, "Advanced real-time mission management for an auv," in *Advanced Mission Management end Sytem Integration Technologies for Improved Tactical Operations, Florence, Italy*, 1999.

[9] J. L. Claude Barrouil, "An integrated navigation system for a long range auv," in *OCEANS'98*, 1998.

[10] AJ. Healey, DB. Marco, RB. McGhee, "Autonomous underwater vehicle control coordination using a tri-level hybrid software architecture," in *IEEE, International Conference on Robotics and Automation*, 1996.

[11] D. Brutzman, M. Burns, M. Campbell, D. Davis, T. Healey, M. Holden, B. Leonhardt, D. Marco, D. McClarin, B. McGhee, R. Whalen, "Nps phoenix auv software integration and in-water testing," in *Symposium on Autonomous Underwater Vehicle Technology,pp.99-108, Monterey California*, 1996.

[12] P. Ridao, M. Carreras, J. Batlle, J. Ama, "Oca: A new hybrid control for a low cost auv," in *Proceedings of the Control Application in Marine Systems*, 2001.

[13] J. Vaganay, B. Jouvencel, P. Lepinay and R. Zapata, "Taipan an AUV for very shallow water applications" in *World Automation Congres*, 1998.