Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems

On AUV Control Architecture

P. Ridao*, J. Yuh+, J. Batlle*, K. Sugihara†

*Informatics and Applications Institute
University of Girona
Avda. Lluis Santaló s/n Girona CP. 17007 Spain.
{pere,jbatlle}@eia.udg.es

+Mechanical Engineering Dept.
†Information and Computer Science Dept.
University of Hawaii at Manoa.
Honolulu, Hawaii 96822, USA
yuh@eng.hawaii.edu

Abstract

This paper surveys control architectures proposed in the literature and describes a control architecture that is being developed for a semi-autonomous underwater vehicle for intervention missions (SAUVIM) at the University of Hawaii. Conceived as hybrid, this architecture has been organized in three Layers: Planning, Control and Execution. The mission is planned with a sequence of sub-goals. Each sub-goal has a related task supervisor responsible for arranging a set of pre-programmed task modules in order to achieve the sub-goal. Task modules are the key concept of the architecture. They are the main building blocks and can be dynamically re-arranged by the task supervisor. In our architecture, deliberation takes place at the planning layer while reaction is dealt through the parallel execution of the task modules. Hence, the system presents both a hierarchical and an heterarchical decomposition, being able to show a predictable response while keeping rapid reactivity to the dynamic environment.

1. Introduction

Modern development in the fields of control, sensing, and communication has made increasingly complex and dedicated robot systems a reality. Used in a highly hazardous and unknown environment, the autonomy of the robots is key to a mission solution. Control architecture is a framework that manages both the sensorial and actuator systems, thus enabling the robot to undertake a user-specified mission.

Since the *Shakey* robot was presented in 1971, a great number of control architectures have been implemented and applied to mobile robots, underwater robots, robots for planetary exploration, and so forth. Different approaches to autonomous underwater vehicle (AUV) control have been discussed in the literature [1, 2, 3, 4, 5, 6]. They are usually classified into three main categories: deliberative, reactive and hybrid.

Deliberative Architecture

Deliberative architectures are based on planning and also on a world model. They allow reasoning and making predictions about the environment. Data flows from sensors to the world model (bottom-up), which is used to plan new actions to be undertaken by the actuators (top-down). When dealing with a highly dynamic environment, the delay in the response time is the main drawback.

This approach is used in the Planning Software Architecture proposed by Hall *et al.* [6]. It is a hierarchical planner arranged into three homogeneous layers. Rock and Wang [11] described an architecture applied to OTTER. It has a three level control structure including a *task level*. Barnett *et al.* [12] used a deliberative architecture called AUVC. It is organized in three hierarchical levels: planning, control, and diagnostic.

Reactive Architecture

Behavioral architectures, also known as reactive architectures or heterarchies, have been discussed in the literature [7, 8, 9]. Decomposition is based on the desired behaviors for the robot and missions are normally described as a sequence of phases with a set of active behaviors. The behaviors continuously react to the situation sensed by the perception system. The robot's global behavior emerges from the combination of the elemental active behaviors. The real world acts as a model to which the robot reacts, based on the active behaviors. As active behaviors are based on the sense-react principle, they are suitable for dynamical environments. Since each behavior pursues its own goal, reaction actions issued by one behavior may cause another behavior to deviate from its respective goal. Then, at times, the robot behavior is not predictable.

The work on reactive architectures was started by Brooks [7] who proposed the subsumption architecture that layers the control system in a parallel set of competence-levels, tying the sensors with the actuators. It uses a priority arbitration through inhibition and/or supression. Bellingham *et al.* [13], adapted the subsumption architecture to the Sea Squirt AUV. They also extended the arbitration mechanism by proposing the *masking*. Zheng [14] introduced the cooperation concept within a subsumption-like control architecture. He also used a layered sensing subsystem dealing with fault tolerance. Payton *et al.* [15] used a behavioral approach to build a

0-7803-6348-5/00/$10.00 ©2000 IEEE.

distributed fault tolerant control architecture. Boswell and Leany [16] applied a layered control architecture to the Eric underwater robot, introducing protected and hormone modules. The Distributed Vehicle Management Architecture (DVMA) was presented by Fujii and Ura [17]. The basic idea is that a behavior can be created by a combination of specific functions given to the robot, and a mission is accomplished by the robot performing sequentially appropriate behaviors. Recently, Rosenblatt *et al.* [18] have applied the Distributed Architecture for Mobile Navigation (DAMN) to the Oberon submersible. Its main feature is the coordination mechanism that is a competence scheme using a voting mechanism to select an appropriate action.

Hybrid Architecture

Hybrid architectures take advantage of the two previous architectures while minimizing their limitations. They usually consist of three layers [10]: (1) the deliberative layer, based on planning, (2) the control execution layer, and (3) a functional reactive layer.

Bonasso [19] described a situated reasoning architecture applied to the Hylas underwater vehicle. Behavior coordination follows a competitive approach and the architecture also includes a deliberative layer. The RBM architecture developed by Healey *et al.* [20] is organized in three levels: *Execution level, Tactic level,* and *Strategic level.* Borrelly *et al.* [21] presented an Open Robot Controller Computer-Aided Design Architecture (ORCCAD). They introduced the concepts of robot-tasks (RT) and module tasks (MT). A mission is built by sequencing RTs and defining event handlers. Borges *et al.* described the DCA architecture [22] that permitted the real-time parallel execution of tasks. It was based on a hierarchical structure consisting of three levels, *Organization, Coordination,* and *Functional* level. Choi *et al.* [23] developed an architecture for the ODIN AUV. It uses a supervisor to handle mission parameters on the basis of lower-level information and three separate blocks: *Sensory database, Knowledge base,* and *Planner.* Recently, Valavanis *et al.* [2] presented the state-configured embedded control architecture which is organized in two-layers: (1) a supervisory control level and (2) a functional control level. This architecture uses a Master Controller (MC) to coordinate the operation of the AUV by transferring control actions among several functionally independent modules.

Summary

Many control architectures recently proposed converge to a similar structure that addresses the use of reusable and modularized software packages such as task modules and behaviors that are linked together for both predictability and reactivity.

The paper is organized as follows: section 2 presents the Intelligent Task-Oriented Control Architecture (ITOCA) that is the high-level control of the SAUVIM vehicle. Section 3 deals with the simulation environment and the results, and section 4 reports the conclusions.

2. The Intelligent Task-Oriented Control Architecture

This section describes the ITOCA being developed for a new semi-AUV, SAUVIM, a research project funded by the U.S. Navy to design and build a semi-AUV for intervention missions. Its dry weight is about 6 tons and design depth is 6,000 m. It has multi-CPUs in VxWorks OS, various sensors, and a robotic manipulator. The SAUVIM is described in [24]. ITOCA is a hybrid control architecture organized into three layers (fig.1):

- Execution: contains sensor and actuator groups.
- Control: contains the vehicle low-level controllers. It is in charge of the non-linear control of the vehicle and the arm.
- Planning: is in charge of the high level control of the vehicle during the mission. It is responsible for the mission planning, execution, and supervision.

Each of these layers is described below.

Execution Layer

This layer is responsible for the interface between vehicle hardware, comprised of the sensor group and the actuator group. Sensors are accessed by the controllers of the upper layer and by the remains of the architecture components. The actuator group is responsible for interfacing with the hardware actuators.

Control Layer

The Control layer has the low-level servo control of the vehicle. For SAUVIM, the Adaptive Learning controller [25] is used to control the vehicle in 6 DOF. Fig.2 shows the low-level controller, where η is the actual position and η_d is the desired position.

Planning Layer

A user delivers a mission to the vehicle by using a Graphical User Interface (GUI). The mission is decomposed by the planner supervisor into a sequence of sub-goals. Each sub-goal has a related task supervisor responsible for arranging the task modules in a suitable configuration for the sub-goal undertaken. Task modules are the main building blocks of our architecture. Each task module is designed to perform a well-defined task and have a solid solution for it. Task modules read input values from sensors, other task modules or the task supervisor, and use the Task-Processing-Function to compute the outputs to be sent to other task modules, or the task supervisor (fig. 3).

Within the task module, fault tolerance is addressed by the fault-tolerance function. Some task modules are related to one low-level controller in the control layer. They have a one-to-one relationship with the physical hardware. The initial values for each controller of each task module are transferred from previous task modules or given by the task supervisor.

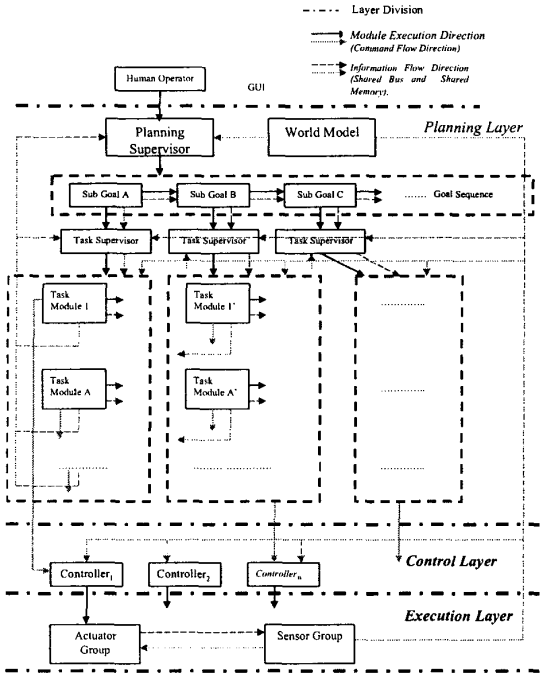
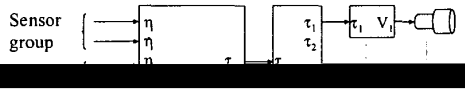


Fig.1 Diagram of the Control Architecture



- Repulsion: avoids obstacles.
- Tracking: follows an object of interest or a sequence of given points.
- Stroll: wanders around.
- Navigation: generates the global path (sequence of way-points).
- Hover: keeps the vehicle position.

Functional task modules are responsible for computing data structures needed by other task modules of the architecture. Some of these modules being worked on include:

- Data-sampling&analysis: samples environmental data for situation detection and exception generation.
- Photographing: takes images about the environment.
- Sleep: waits until new command.
- Local Map Building: builds a local map of the vehicle surroundings.
- Weight Average: merges the output of the enabled Motion task modules into a unique set point to be sent to the low-level controller.

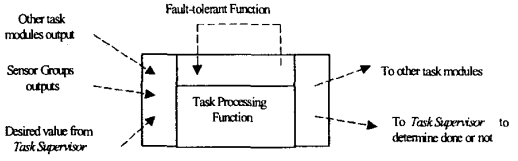


Fig. 3 Task modules

Sub-goals

Sub-goals for AUVs include Docking, Survey, Searching, and Station Keeping. Organization of the task modules for each of these sub-goals includes:

- Docking={Navigation; Repulsion, Local Map Building, Sleep, Approaching/Departing routine, Weight Average}: approach to a docking position for data transmission or recharging the battery.
- Searching={Navigation; Repulsion; Local Map