A HIERARCHICAL CONTROL ARCHITECTURE FOR MOBILE OFFSHORE BASES

Anouck Girard and J. Karl Hedrick
The University of California at Berkeley[*]
João Tasso de Figueiredo Borges de Sousa
The University of Porto, Portugal[**]

ABSTRACT

A hierarchical architecture for Mobile Offshore Bases (MOB) control is presented. By a control architecture we mean a specific way of organizing the motion control and navigation functions performed by the MOB. It is convenient to organize the functions into hierarchical layers. This way, a complex design problem is partitioned into a number of more manageable sub-problems that are addressed in separate layers. The decomposition also allows for modular design and testing and the incorporation of plug-and-play components.

This paper discusses the MOB requirements and maps them onto a layered control architecture. The formalization of the hierarchy is accomplished in terms of the specific functions accomplished by each layer and of the interfaces between layers. The implementation of the layers is discussed and illustrative examples are provided.

Keywords: Mobile Offshore Base, MOB, Hybrid System, Control Architecture, Supervisory Control, Dynamic Positioning

1. INTRODUCTION

The concept of a floating, at-sea base stems from the necessity for the United States to be able to stage military and/or humanitarian operations in any part of the world. A Mobile Offshore Base (MOB) is a large self-propelled, floating, pre-positioned ocean structure formed of three to five platforms and reaching up to 1,500 meters in length. The MOB must be able to accommodate C-17 aircraft operations, and cargo transfer from container ships. At this stage, feasibility remains a fundamental question, and control systems and dynamic positioning technology must be developed and evaluated. This effort is part of a larger project, as described in [1]. The CASSETTE (Development And Demonstration Of Control And SyStem EvaluaTion TEchniques) project from PATH - UC Berkeley is part of the MOB technical base effort devoted to determining the feasibility of dynamic positioning of multiple MOB platforms. In this project we are developing an automated Multi-Module Dynamic Positioning Control System (MMDPCS) for the MOB, and a simulation template to uniformly support MMDPCS designs and performance evaluations.

Under this project the team was tasked to virtually demonstrate the MOB control and evaluation method, and physically validate the key design issues with scale models of the MOB.

In this paper, we propose a hierarchical control architecture for the control and maneuver coordination of multiple modules within the MOB. Example maneuvers include, forming or breaking connections between MOB components, and reorientation of the entire MOB string in the direction of the wind. The proposed architecture distributes information and control authority among modules and deals with exceptions and faults. It is organized hierarchically, with the lower layers of the hierarchy consisting of continuous controllers that interact with the sensors and actuators to produce the desired positioning and tracking performance. Higher layers of the hierarchy will be modeled by discrete event systems used for maneuver coordination, fault identification and reconfiguration. These layers sequentially organize the generation of the optimal coordinated trajectories for all modules (global control), the optimal trajectories for each module (module control), the optimal trajectory

[*] 5141 Etcheverry Hall, University of California at Berkeley, Berkeley, CA 94720, USA
anouck@robotics.eecs.berkeley.edu, khedrick@me.berkeley.edu
[**] FEUP, Faculdade de Engenharia da Universidade do Porto, R. dos Bragas, 4099 Porto Codex, Portugal
sousa@robotics.eecs.berkeley.edu

tracking schemes and the optimal motor controls (local control), to name just a few. The de-coupling of optimization problems eventually compromises the overall optimality but makes the problem tractable. Hence, the architecture represents an empirical compromise between tractability and optimality. It will become apparent that the task is formidable, even in a reduced setting. Here we give an overview of what is involved, describe the key design concepts and establish a framework for tackling the problem.

This paper is organized as follows. In section 2, some of the representative requirements for the MOB are presented, along with some expected maneuvers and mission profiles. In section 3, we outline the proposed solution and describe the control architecture that is the starting point of our design. In section 4, we briefly discuss the implementation of the architecture in the MOB SHIFT framework. Controller design and implementation are described in section 5. Finally in section 6, we highlight some of the subtle issues raised by the design process.

2. MOB REQUIREMENTS

2.1 Terminology

The words "platform" and "module" are used interchangeably. A *Module* is a semi-submersible platform that is used as a component for the MOB. *MMDPCS* represents a module's Multi-Module Dynamic Positioning Control System. A *MOB* is defined as a set of independent modules providing landing capabilities for conventional take-off and landing of aircraft. *Up Time* is the fraction of the mission duration when the MOB is available for air operations. The term "*docking*" was used to represent the action of assembling two modules end-to-end.

2.2 Modes of operation

One of the MOB mission requirement under analysis is the capability to support sea and air operations. In order to achieve this goal the MOB is required to:

- Be aligned with the dominant wind, local currents, waves, etc. that may introduce transverse perturbations.
- Remain operational for CTOL (conventional take-off and landing) aircraft through Sea State 6.
- Disassemble to survive worse environments, wind and wave drift forces, deriving from Sea State 7.
- Reassemble when the environment reduces to Sea State 6.

The operational availability for air operations is the single most important measure of effectiveness of this capability. The anticipated modes of operation are as follows:

- Unassembled mode: each individual module maintains a prescribed position and orientation.
- Assembled mode: the MOB assembly maintains a general position and orientation, and the individual modules maintain relative positions and orientations.
- Transitioning modules: independent modules are in preparation for mechanically connecting/disconnecting the modules end-to-end. Each transitioning mode corresponds to some prescribed sequence of operations. There are two major transitioning modes:
  - the normal mode where the assembling and disassembling operations are executed one platform at a time, and the
  - emergency mode, where disassembling is executed as fast as possible under survival conditions deriving form Sea State 7.

The MOB operation requires adaptation to the environmental conditions. By adaptation we mean the downgrading of some specifications to be able to fully accomplish the operational requirements. A lattice of preferred operating modes is provided as a constraint that will be used to trigger the required adaptation schemes. This lattice does not include fault-recovery modes that are treated separately in the architecture.

2.3 Mission profiles

An example of a typical mission is presented below.

In a first time, several independent modules assemble to form a MOB. This requires position and speed controllers for each module, a dynamic positioning controller for the immobile module(s) and a docking controller for the approaching module.

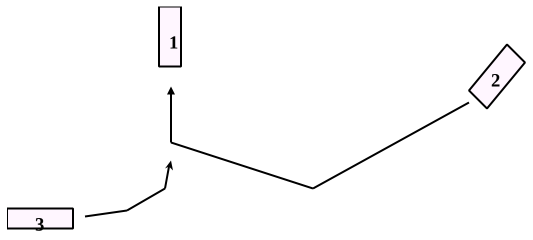


Figure 1.
From unassembled mode to assembled mode.

Then, the group goes to and maintains a desired position while aligning with the wind. After a given period of time, the MOB breaks-up and the modules return to independent operation. This phase assumes that the assembled MOB will be able to perform dynamic positioning, alignment into the wind and wind tracking. Also, individual modules need speed and position control for the disassembly phase.

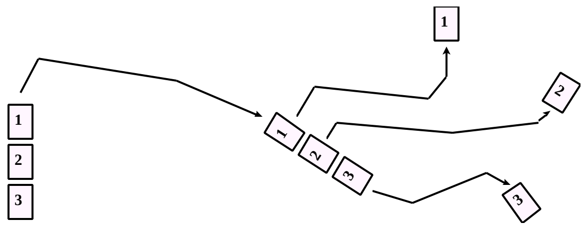


Figure 2.
From assembled mode to unassembled mode.

Examining this profile, we extracted a finite, exhaustive number of maneuvers that can be used to form all MOB missions. Should the need for additional maneuvers arise, the architecture is sufficiently flexible to accommodate them with minimal changes.

3. OUTLINE OF PROPOSED SOLUTION

Due to their unusual size and weight, very large floating structures in general, and Mobile Offshore Bases in particular, raise some specific control issues. The description requirements are very complex since the whole system includes both continuous activities and discrete-event features (i.e., constitutes a hybrid system) evolving in several time scales. The dynamic nature of the problem stems from the existence of multiple vehicles whose roles, relative positions, and dependencies change during operations. To meet these complex system description requirements, the architecture is modeled as a dynamic network of hybrid systems using the SHIFT [9] formalism and specification language.

3.1 Assumptions

For the purpose of MMDPCS design and performance evaluation, the MOB system comprises the following elements:

1) Hierarchical control architecture.
2) Independent systems:
    a) Semi-submersible platform.
    b) Power plant.
    c) Thrusters.
    d) Sensors.
    e) Flexible bridges (de-couples motion from other platforms in the MOB).
    f) Connectors (couples the motion with that of other platforms in the MOB).

The following assumptions were considered in the design of the MOB SHIFT Evaluation Framework [12].

1) Fixed number of modules.
2) Thruster and relative position sensor failures are the only physical failures modeled in the MOB SHIFT simulation framework.
3) Relative position sensor failures are the unique cause for drive-off.
4) The connectors and bridges are 100% reliable. However, the current version of the hierarchical control architecture does not prevent stress beyond a prescribed limit. Hence, induced physical failures may occur.
5) The current version of the hierarchical control architecture does not include fault detection and recovery schemes, except for the thrust allocation unit that accommodates some types of failures.
6) All faults are observable.

3.2 Design Process

The control architecture design will consist of the following steps:

1) Deciding on the number of levels, their role, their descriptive language and the way they interact.
2) Identifying the information that is needed at each level to make meaningful decisions.
3) Designing interfaces to the sensing and communication architecture.

The design process benefited from the PATH experience in the motion coordination of multiple vehicles [2]. The PATH architecture design organizes individual motions into a small number of prototypical maneuvers that are used as atoms for more complex prototypical maneuvers involving the motion coordination of several vehicles. Prototypical motions can then be verified for safety and consistency as required by the safe operation of the whole system. This same organization was mapped onto the formal layered PATH architecture.

Although less structured than an automated highway, the control and coordination of the MOB can be decomposed according to the same principles. The movement of modules is realized through basic maneuvers (as described below) that are coordinated.

To complete a mission as described in section 2.3, five separate controllers are necessary:

- go to a specified location (move MOB),
- assemble with other modules (assemble MOB),
- dynamically position (DP MOB) and align in the wind (align MOB in wind),
- split up MOB.

Hence there are five different specific maneuvers for each module to perform. These maneuvers, in turn, require the consideration of a finite number of atomic control laws:

a) The "go to a point" law allows the module to move to a specified location.
b) The "trajectory tracking" law is a more refined version of the previous one, which allows you to track a trajectory. This is particularly useful for assembling and splitting up the modules of a MOB.
c) The "decelerate to assemble" law is used to control the speed of the platform accurately

while assembling. This is necessary to avoid collisions and increase the precision of the assembling maneuvers.

d) The "dynamic positioning" law is used to maintain the alignment of the platforms and control their inertial and relative positions.

e) The "leaderless", "first as leader", "middle as leader" and "follower" laws all refer to different schemes to distribute the control between platforms. These are described in more detail in [7] and in section 5.1.1.

After identifying the atomic laws, the requirements in terms of actuator, sensor and communication capabilities were laid out, as shown in figure 3.

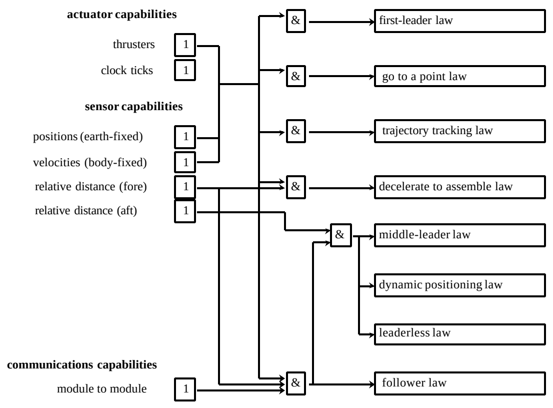


Figure 3.
System requirements for atomic laws.

The atomic laws can be combined to form all the basic maneuver laws, as illustrated in figure 4. A collision avoidance law is presented to make the set complete, but has not been implemented or tested yet.

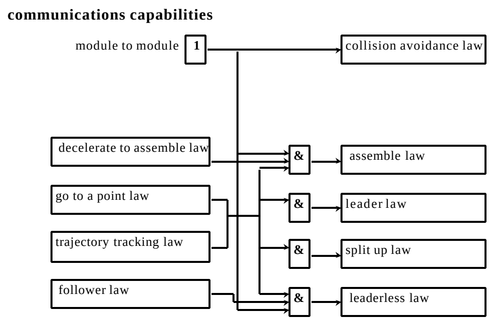


Figure 4.
Combination of atomic laws to obtain basic laws.

Each basic maneuver is paired with a compatible reference generator.

At this point, a controller was designed for each of the basic maneuvers. Each basic maneuver requires a hybrid controller that coordinates the execution of the underlying atomic laws and the corresponding reference generator according to the logic encapsulated in a state machine. The controllers and reference generators all use the laws described in figures 3 and 4.

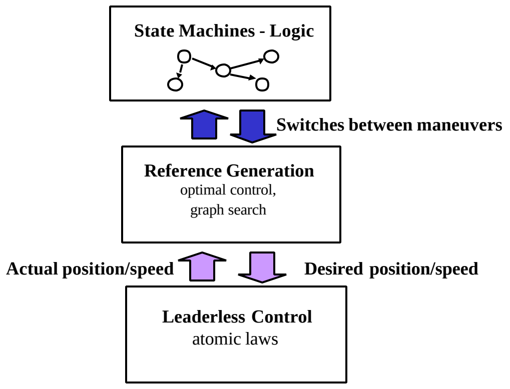


Figure 5.
Organization of basic maneuver control.

The dependencies for each of the five aforementioned controllers are listed in figure 6. Identifying the dependencies of each controller on communications, sensors and actuators is particularly useful in the context of fault tolerant architectures. It allows for reconfiguration of the system in the presence of failure, for example having two functioning platforms following one with multiple thruster failures to remain operational.

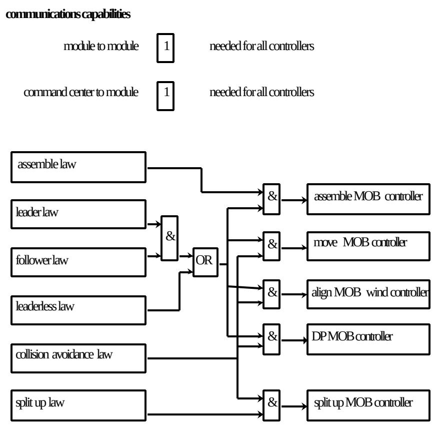


Figure 6.
System requirements for controllers.

3.3 Overview of Architecture

To deal with mission handling and control as well as dynamic positioning of the platforms and safety issues, a three-layer software architecture (as presented in figure 7) that moves from discrete to continuous signals was used [2,3,4]. It should be noted that our design is not necessarily unique or optimal, but as a preliminary approach is sufficient to prove our point. Different alternatives for hybrid system design can be found in [5,6].

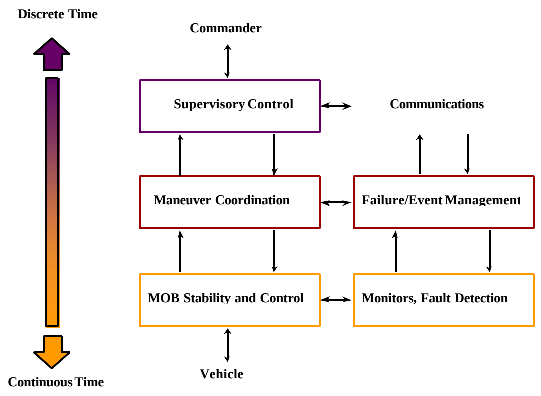


Figure 7.
Hierarchical Control Architecture for the MOB.

- **MOB stability and control layer:** the automated modules. The module dynamical models are given in terms of nonlinear ordinary differential equations. This level deals with continuous signals, and interfaces directly with the platform hardware. It contains several dynamic-positioning algorithms, a thruster allocation scheme, and sensor data processing and monitoring for fault detection. Control laws are given as vehicle state or observation feedback policies for controlling the vehicle dynamics. Sensor processing at the MOB stability and control layer includes environmental monitors that are used to signal changes in the environmental conditions. The corresponding events are sent to the maneuver coordination layer that will promote the change to the next preferred mode.
- **Maneuver coordination layer:** control and observation subsystems responsible for safe execution of atomic maneuvers such as assemble, split, wind tracking, and go to a location. Maneuvers may include several modes according to the lattice of preferred operating modes. Mode changes are triggered by events generated by the stability layer monitors. This feature provides a first level of dynamic reconfiguration that is further extended to accommodate fault handling. The coordination layer 's primary concerns are to deal with configuration changes of the MOB and to optimize its alignment and fuel consumption, without putting the platforms at risk. With these criteria in mind, it computes optimal positions for each platform. It also monitors incidents and reacts to minimize their impact on maneuvers and maximize safety.
- **Supervisory control layer:** control strategies that the modules follow in order to minimize fuel consumption and maximize safety and efficiency. Discrete commands are given to achieve high-level goals of overall coordination and maneuvering of the mobile offshore bases. This layer monitors the evolution of the system with respect to global mission goals. It receives commands and translates them into specific maneuvers that the platforms need to carry out. In the current design, finite state machines are used to represent the communication protocols and organize them in a systematic way.
- **Interfaces between layers:**

a) Vehicle: the platforms input actuator commands (can be desired thrust and orientation for each thruster or direct voltage commands to thruster motors) from the stability and control layer, and output information on whether the thrusters and sensors are functioning properly.

b) MOB stability and control layer: the MOB stability and control layer inputs set points in either position or speed from the maneuver coordination layer, and outputs state information to the maneuver coordination layer.

c) Maneuver coordination layer: the maneuver coordination layer inputs specific maneuvers to execute from the supervisory control layer, and outputs state information and information about the status of the current on-going maneuver to the supervisory control layer.

d) Supervisory control layer: the supervisory control layer inputs commands and outputs all state and on-going maneuver information as well as information regarding whether the global mission goals are being met.

e) Commander: the commander inputs all available state, maneuver and mission information and outputs high-level commands for the MOB to execute.

4 MOB SHIFT SIMULATION

The structure of the SHIFT simulation environment is as presented in figure 8.

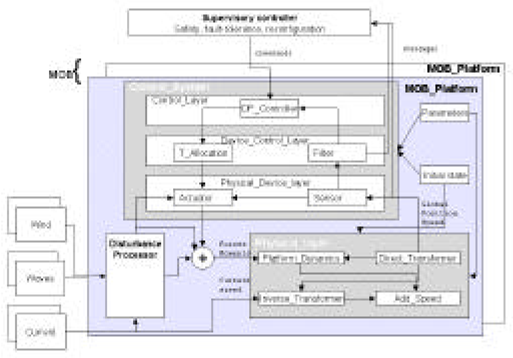


Figure 8.
Organization of the MOB SHIFT architecture.

Each MOB is equipped with a control system, a physical layer containing a physical model of the platform and a disturbance processor. Each component and its underlying model is described in detail in [10], as well as the connections between components.

5 CONTROLLER DESIGN

In this section we give a brief description of the multi-layered control design that is necessary for mission control of a MOB. Each layer will be examined individually. The set of controllers described here is not unique; it corresponds to the particular approach developed by the University of California, Berkeley. In this section we present a sketch of the controller designs for this control architecture, with special emphasis on dynamic positioning, one of the major thrusts of the CASSETTE developments. This architectural concept was partly implemented in SHIFT as part of the MOB SHIFT evaluation framework. The controller interfaces can be reused to incorporate other dynamic positioning controllers into the MOB-SHIFT framework, so comparisons can be made under the same set of assumptions.

5.1 MOB stability and control layer

The primary focus of the MOB stability and control layer is to ensure smooth operation of a single module and to inform the coordination layer of whether set points are reached or not.

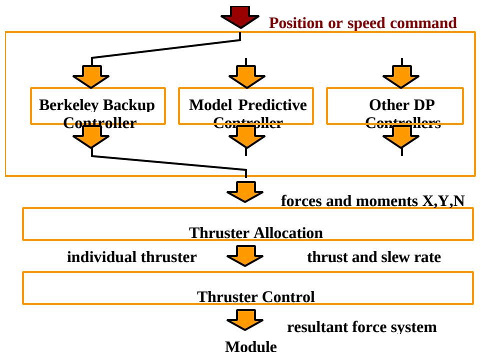


Figure 9.
Organization of the MOB stability and control layer.

The organization of the stability and control layer is presented in figure 9. It contains several different dynamic positioning algorithms and a thruster allocation scheme.

5.1.1 Dynamic Positioning (DP) algorithms

Two different dynamic positioning algorithms are being developed and tested using MOB-SHIFT.

5.1.1.1 Berkeley Backup Controller (BBC)

The CASSETTE team at the University of California, Berkeley has developed a non-linear DP controller for MOBs. The approach uses multiple sliding surface control, dynamic surface control, and the Slotine and Li algorithm [13] and can be implemented in either robust or adaptive form. Detailed equations and control laws and surfaces are given in [7]. The controller is used in different forms to implement all of the several atomic laws as described in section 3.2 and figure 3. It inputs a position or a speed command, and outputs a force system to be applied on the module. The controller also uses information sent by the maneuver coordination layer that regards which multiple module scheme to use (if any), and whether to use integral control or not.

The controller can be used in several coordination schemes to coordinate the movements of multiple platforms.

- The first concept tested was a “follow-the-leader” scheme were the first module in the string is designated as leader, and is controlled to track a desired inertial position (x, y, ψ). The second and third modules respectively track the first and second module. In other words, each module “sees” only the module located directly in front of him. A drawback of this approach is that “string stability” problems may appear, i.e. spacing errors may propagate and amplify along the string, potentially causing a collision [11].

- The second concept to be evaluated was a “follow-the-leader” scheme where the middle module was chosen as leader. String stability is not a problem for a three-module string because the end modules directly track the leader, and is less of a problem for longer strings because the string is half as long on each side.

- The final concept to be tested was a leaderless control scheme where each module is controlled to track both an inertial reference and its position relative to neighboring modules.

As shown in [7], the leaderless control scheme was able to maintain the fastest alignment of platforms and the better alignment performance in time. The controller was tested for a variety of disturbances and environments.

An example of a typical dynamic positioning test run is given in figure 10. Plots show movements in x, y and psi versus time. The platforms are displaced form their original position and return to their desired position. There is no overshoot or steady state error.

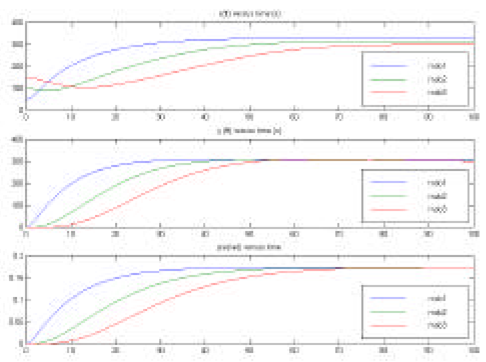
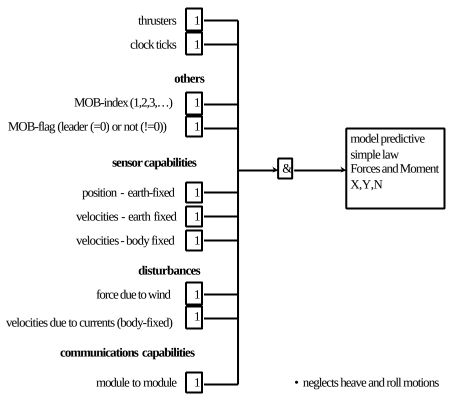


Figure 10.
Example DP simulation run.

5.1.1.2 Model Predictive Controller (MPC)

A model predictive controller was developed by SSCI, and is described fully in [8]. In terms of interfaces and dependencies, the controller is described in figure 11.



thrusters
clock ticks
others
MOB-index (1,2,3,…)
MOB-flag (leader (=0) or not (!=0))
sensor capabilities
position - earth-fixed
velocities - earth fixed
velocities - body fixed
disturbances
force due to wind
velocities due to currents (body-fixed)
communications capabilities
module to module
&
model predictive simple law Forces and Moment X,Y,N
• neglects heave and roll motions

Figure 11.
Dependencies for the MPC simple control law.

5.1.2 Thruster allocation scheme

An allocation scheme for multiple thrusters was developed. The approach is based on linear programming. It allows us to consider many of the real-world thruster characteristics, such as maximum thrust capability and thrust rate, maximum slew rate of the thrusters etc.. The thruster allocation scheme inputs a desired force system for a given module, and thrust and azimuth commands are computed for each individual thruster so that energy consumption is minimized and the desired force system for the module is achieved. The approach was programmed and tested, and full equations and details of the computation as well as results appear in [14].

5.2 Maneuver coordination layer

The maneuver coordination layer interfaces high-level supervisory control (discrete) with low-level continuous control of the module. It contains several controllers as well as some logic for switching between controllers. The five controllers in the maneuver coordination layer (dock, split, go to, align in wind and DP) send either speed or position set points to the controller described in section 5.1.1.1.

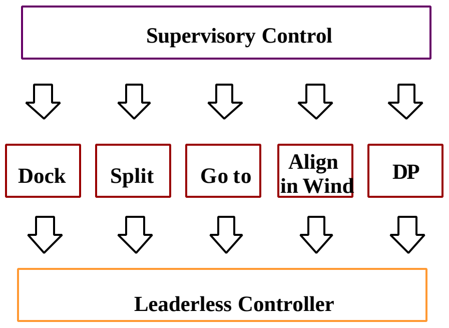


Figure 12.
Organization of the maneuver coordination layer.

Each controller is formed of a control law, and a "*protocol*" that is used to coordinate maneuvers. The current design uses protocols in the form of finite state machines to organize the maneuvers in a systematic way. They receive the commands from the supervisory controller and aggregated information from the individual platforms, then use this information to decide on a control policy and issue commands to the MOB stability and control layer.

As an example, the protocol for the docking controller is given here. It is formed of two state machines, one for each module involved.

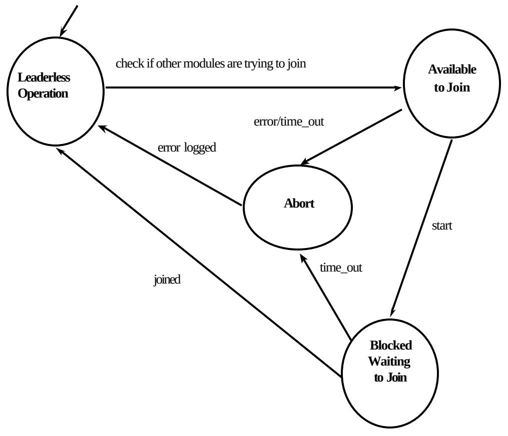


Figure 13.
State machine for immobile module.

For simplicity, we assume only one module is moving during docking. We call this module the "mobile module". The state machine containing the logic for the mobile module is presented in figure 14. The other module must be in dynamic positioning, and as a simplification is assumed to be “immobile”. The state machine containing the logic for the immobile module is presented in figure 13.

The immobile module gets a request asking it whether it is available to assemble, if not a time out occurs and the maneuver is aborted and if yes “blocks” itself so that no other module may dock while the maneuver is taking place.

When the maneuver is complete, the module goes back to leaderless operation, and is once again available to assemble.

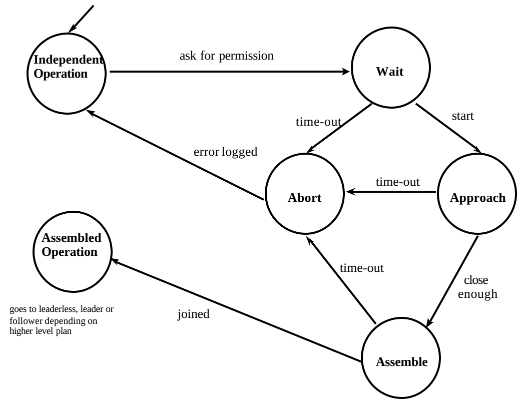


Figure 14.
State machine for mobile module.

The mobile module asks the immobile module for permission to dock. If it is not granted, the module goes back to independent operation, otherwise the module starts its approach towards the immobile module. It is aimed at the closest corner of a 3000m grid centered about the immobile module. When the mobile module enters the grid, a fuzzy logic system that computes desired velocities is activated.

When the platforms have assembled, the MMDPCS is activated, the immobile module is made available for other modules to assemble themselves.

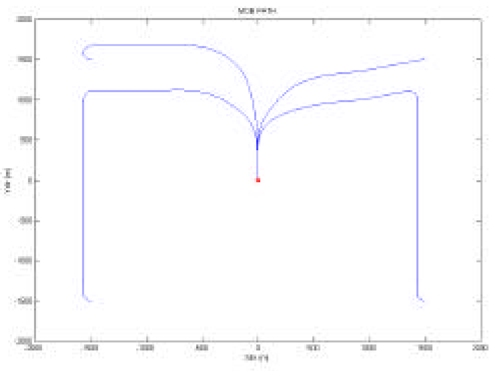


Figure 15.
Preliminary results (in Matlab) for docking controller.

Preliminary results for the fuzzy-logic docking controller are shown in figure 15. It shows the trajectory of the mobile module for four different runs, one starting at each corner of the grid. Originally, the mobile module is traveling in a direction that is 280 degrees from that of the immobile module with a speed of two knots. The docking controller has yet to be ported to MOB SHIFT.

5.3 Supervisory control layer

The supervisory control layer resides on top of the architecture and is concerned with the planning and execution of the MOB mission.

The supervisory controller commands the maneuver layer to execute a sequence of proper behaviors, which enable the MOB to accomplish the overall mission. It also allows for the coordinated operation of several MOBs.

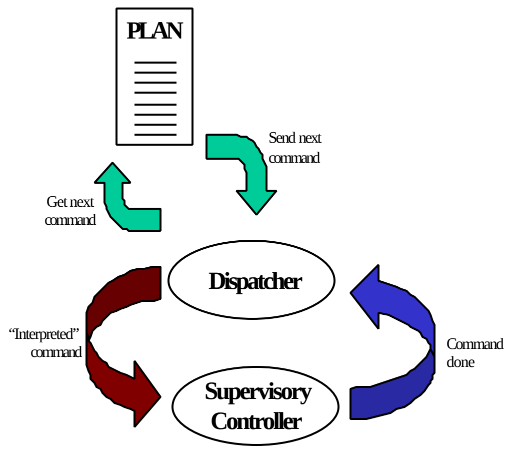


Figure 16.
Mission plan execution.

Commands are specified in a mission plan, that contains instructions such as “do dynamic positioning with modules number 2, 4, 3, 1, 7 at position x, y, ψ for time t” as presented in figure 16.

The dispatcher then reads the commands and reformulates them so that the supervisory controller can interpret them. It also acknowledges whether commands have been completed or aborted as indicated by the supervisory controller. A sketch of the supervisory controller is presented in figure 17.

The MOB can be in two states, a default state named “*Idle*”, and the “*MOB*” state where the MOB is active. In this design, four commands can be given. The “*Start*” command that adds the first module to the MOB, a “*Join*” command for adding additional modules, a “*Split*” command that removes modules from the MOB, and an “*Idle*” command that returns the

MOB to idle if necessary. An "emergency split" command can also be implemented to split up (disconnect and send to a reference point) all of the modules forming a MOB at the same time.

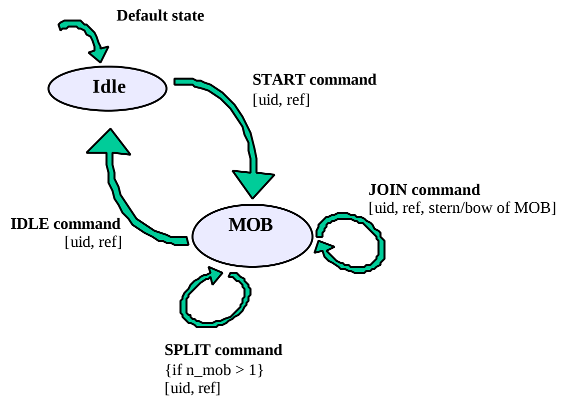


Figure 17.
MOB state machine.

We see that each layer of the control hierarchy involves different entities and algorithms that relate entities at one layer to entities at the adjacent layers. Moreover, there is a theoretical framework at each layer which provides a meaning to its entities, and relations between those frameworks that are operational in terms of those algorithms.

6 CONCLUDING REMARKS

In this paper we have proposed a hierarchical control architecture for the MOB. A hierarchical structure for control and planning has been adopted to provide a systematic design framework for this complex system. SHIFT, a hybrid systems programming language developed at California PATH, was used to develop a virtual simulation tool in which the architecture was tested and evaluated. A reduced version of the architecture will support the physical validation of the key design issues with three scale models (1:150) of the generic MOB. The implementation will be done in TEJA, an object-oriented framework for implementing real-time, event-driven, distributed multi-agent control systems, from Teja Inc.

Acknowledgements: The material is based upon work supported by the U.S. Office of Naval Research's MOB Program under grant number N00014-98-1-0744. The authors would like to thank Prof. William Webster, Jim Misener, Prof. Pravin Varaiya and Akash Deshpande for helpful discussions providing insight into this problem.

References

[1] R. Zueck, P. Palo and R. Taylor, "Mobile Offshore Base: Research Spin-Offs", Proc. Of the ISOPE Conference, Brest, France, June 1999, pp 10-16.

[2] P. Varaiya, "Smart Cars on Smart Roads: Problems of Control", IEEE Trans. Of AC, Vol 38, No. 2, February 1993

[3] F. Eskafi, D. Khorrambadi, and P. Varaiya, "SmartPath: An Automated Highway System Simulator", Tech Rep PATH Tech Memo 92-3, Institute of Transportation Studies, University of California, Berkeley, CA 94720, October 1992

[4] A. J. Healey, D.B. Marco and R. B. McGhee, Autonomous Underwater Vehicle Control Coordination Using A Tri-Level Hybrid Software Architecture, Proc. of 1996 IEEE Conference on robotics and Automation, Minneapolis, Minnesota, April 1996, pp 2149-2159.

[5] A. Girard, "A Convenient State Machine Formalism for High-Level Control of Autonomous Underwater Vehicles", Master's Thesis, Florida Atlantic University, Boca Raton, FL 33431, May 1998

[6] D. N. Godbole, J. Lygeros and S. Sastry, "Hierarchical Hybrid Control: An IVHS Case Study", in Hybrid Systems II (P. Antsaklis, A. Nerode and S. Sastry, eds.), no 999 in LNCS, Springer Verlag, 1995.

[7] K. Hedrick, A. Girard and B. Kaku, "A Coordinated DP Methodology for the MOB", Proc. Of the ISOPE Conference, Brest, France, June 1999, pp 70-75.

[8] V. Manikonda, M. Gopinathan, P. Arambel and R. K. Mehra, "Nonlinear Model Predictive Control Design for Coordinated Dynamic Positioning of a Multi-Platform Mobile Offshore Base", accepted for the VLFS'99 Conference, Honolulu, Hawaii, September 1999.

[9] www.path.berkeley.edu/shift/

[10] J. Sousa, N. Kourjanskaia, and A. Girard, "The MOB SHIFT Simulation Framework", accepted for the VLFS'99 Conference, Honolulu, Hawaii, September 1999.

[11] D. Swaroop and J. K. Hedrick, "String Stability of Interconnected Systems", IEEE Transactions on Automatic Control, March 1996, Vol. 41, N3, pp 349-357

[12] "Technical Progress Report #3", March 1999,

CASSETTE Project, University of California at Berkeley, Partners for Advanced Transit and Highways.

[13] J.J.E. Slotine and W. Li “Applied Nonlinear Control”, Prentice Hall, Englewod Cliffs, NJ, 1991.

[14] W.C. Webster and J. Sousa, “Optimum Allocation for Multiple Thrusters”, Proc. Of the ISOPE Conference, Brest, France, June 1999, pp 83-89.