A COMMON CONTROL LANGUAGE FOR DYNAMIC TASKING OF MULTIPLE AUTONOMOUS VEHICLES[1]

Christiane N. Duarte duartecn@npt.nuwc.navy.mil
Gerald R. Martel martelgr@npt.nuwc.navy.mil
Naval Undersea Warfare Center
Newport, R.I. 02841

Eugene Eberbach eeberbach@umassd.edu
Christine Buzzell g_cbuzzell@umassd.edu
University of Massachusetts Dartmouth
Computer and Information Science Department
Graduate School of Marine Science and Technology
North Dartmouth, MA 02747-2300

Abstract

Cooperative behavior, visible in many animal species, is achieved through varying degrees of communication (e.g. chemical clues from ants, a bees "dance"). When looking at military operations, groups communicate. In the networking of computers and their coordination, communication is necessary to achieve goals. In the same way, cooperative behavior for autonomous vehicles will require some level of communication between group members. Both military and scientific concepts for autonomous operations are projecting the use of a variety of heterogeneous platforms in group-operations in order to address various operating regimes and specialized tasks thus introducing unique issues for coordinated operations.

We will provide an overview of a Common Control Language (CCL) approach that is a natural extension of a behavior-based control although not limited to such an implementation. In our efforts to define and design a CCL, we are interested in answering the following questions: Can autonomous group control benefit from a specialized control language? Can we capture a language to support agent coordination? Our approach will center on 3 critical requirements that this language must address: 1) a process- descriptive language to be used by the scheduler of a vehicle controller, 2) a vocabulary that can evolve and grow with new applications but is primarily based on a discrete set of fundamental behaviors and 3) a translation scheme for an operator's mission specifications.

Introduction

[1] **Research partially supported by grants from Office of Naval Research ONR N66604-03-M-4659 and ONR N000140310421**

Unmanned systems are an area of growing war-fighter interest due to their potential as force multipliers, especially against asymmetric threats. Furthermore, swarms or collaborative groups of unmanned vehicles are seen as revolutionary and transformational technologies for war-fighting and homeland defense. Future concepts for autonomous operations are projecting the use of heterogeneous platforms in group-operations in order to address various operating regimes and specialized tasks thus introducing unique issues for coordinated operations. In addition operator and group members are more effective when some level of communication can exist between agents. This has motivated the investigation in [1,2,3] of a common control language (CCL) to establish both a standard for communicating between different agents (i.e. underwater vehicle, surface vehicle, human operator) during operations and support the interactions between operator and machine. CCL must meet the needs of an operator managing a group of vehicles or a single vehicle. CCL must allow concise representation of goals or tasks by autonomous agents.

Our approach will center on 3 critical requirements that this language must address: 1) a process- descriptive language to be used by the scheduler of a vehicle controller, 2) a vocabulary that can evolve and grow with new applications but is primarily based on a discrete set of fundamental behaviors and 3) a translation scheme for an operator's mission specifications.

Much related work has been done in computer networks and this will be a model for us. Distributed software agent designers are grappling with similar coordination issues and looking for solutions beyond contract net protocols.

What is a Common Control Language?

We can start with our definition of a CCL as:

> *A concise and robust language to be used by autonomous agents for communicating information and coordinating tasks*

We clarify further the term autonomous agent by defining some minimal characteristics:

- Can "understand" a set of task specifications
- Is situationally-aware
- Can make decisions

An agent can recognize a task as within its repertoire and take responsibility to perform the task or respond by ignoring or flagging to others that it is unfamiliar with this task. During the entire mission, the agent is able to maintain some degree of awareness of its surroundings depending on the accuracy and resolution of its sensor suite. The level of awareness should be detailed enough to recognize key events related to the tasks it can perform; and critical to the "autonomy". An agent should be capable of making decisions even if rudimentary.

We are planning to adhere to the following overall design goals for the implementation of a common control language:

- One vehicle can not look inside another vehicle- coordination is through message-passing
- It should allow for arbitrary execution of behaviors, e.g. repetition, sequential or parallel
- It should be easily extensible to new vehicles and new missions
- It should be possible to use for both real and simulated vehicles
- Users should be able to add their own messages if required but not expect that these new messages will be understood or achievable by all vehicles

Like a Computer Network

With these definitions in mind we can make an analogy of a group of autonomous agents to a network of computers where the role of a CCL can be compared to a computer instruction set. There are basic instructions that a computer can support and a programmer combines these instructions in sequential, parallel and iterative constructs. This has evolved from assembly to high-level language such as C and Java. CCL will also have basic set of instructions or behaviors that we define. They are the building blocks for autonomous vehicle interactions.

In reactive-based controllers behaviors are downloaded and used in a pre-determined fashion. What we propose is a language that will allow autonomous agents to organize and combine these behaviors to task a vehicle or group of vehicles in real-time, based on the situation. Currently there is no standard for such a language. In general any message passing between two or more agents have consisted of pre-defined handshakes of data or triggers specific to a single task not a general approach as we are suggesting.

Coordination

The CCL language has many constructs that will support tasking of a mission from the perspective of the group therefore allowing group coordination. The constructs are primarily two-fold: cost functions embedded in the language and the use of optimization techniques during the translation of the mission.

We look to the methodologies used in operating systems and computer networks to provide anytime dynamic mission definition. We plan to use the well-established methodologies of anytime algorithms for behavior selection when compiling or interpreting a mission.

A vehicle should be responsive based on time and resources. Much like a compiler will have intermediate measures to optimize register use and memory access, we are looking to insert optimization measures when defining the tasks necessary for the mission and their execution (e.g. parallel, sequenced, etc.). Anytime algorithms are solutions designed with consideration to the trade-off between performance and time. As time increases the quality of the output will increase. During autonomous vehicle operations, time can be a limitation but a solution is always necessary.

The CCL module can construct its output by optimizing several steps ahead. CCL optimization algorithms can run behavior sets and provide results on

performance. Thus we have a smart CCL module.

One of our modes of operation will allow for real-time updates to a CCL interpreter from the vehicle and group members. As illustrated ing Figure 1, the CCL interpreter can optimize several steps ahead, send out the tasks, and then update the tasks when real-time status is provided.

This has implications to many situations. As part of a group of vehicles, each member has several interesting potentials. When tasked, a vehicle through its CCL module may determine that it can do the task, it cannot do the task, or joined with another vehicle it can do the task. Thus a coordinated group solution incorporates the resources available not only to the single vehicle but also to

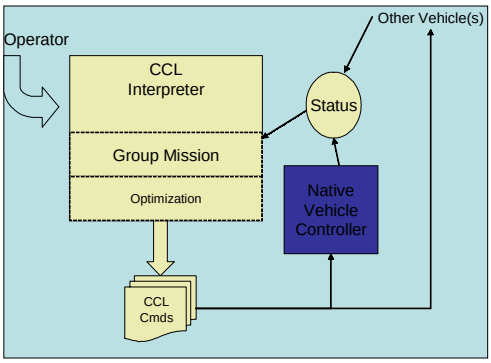

Figure 1: Real-time updates to CCL

the group. It allows coordination between group members when executing a task or dealing with an event.

Language Formalism

Process algebra which comes from the computer science community has been used to describe systems consisting of multiple parallel components and their behavior (e.g., operating systems, computer networks) and is our starting point for the language formalism. Our motivation for starting with process algebra is the similarities in distributed robotics to parallel processing. We feel much can be leveraged from process algebra for multi-agent communication.

To date work has been done in [4,5,6] on developing a language using $-calculus (pronounced cost calculus) process algebra for Unmanned Underwater Vehicles. We are following this line of thinking and taking the steps to implement a prototype.

A Compiler and Interpreter

To develop the CCL module an interpreter and/or compiler will need to be written. Taking the syntax from Eberbach's description of $-calculus the grammar for the CCL will be created. The grammar is a formalized description of how the CCL will function. The CCL module will be written in the C programming language.

- Send / Receive
  (-> a P) send P through channel a
  (<- a X) receive X from channel a
- Cost
  ($ P)
- User Defined Atomic Expressions (AUV elementary behaviors)
  (a P) where a is the name and P a list of parameters
- Cost Choice
  (? P) ex: (? *Continue Return*)
- General Choice
  (+ P) ex: (+ Transit Report Sensing)
- Sequential composition
  (. P) ex: (. Transit Search Return)
- Parallel Composition
  (|| P) ex: (|| Search Monitor)
- Recursive function call and its definition
  - (f X) function f with parameters X
  - (:= (f X) P) recursive definition of f with parameters X and body P

Table1: Syntax for Language

For portability reasons, CCL will be implemented, similar to Java, in two steps: as a compiler and an interpreter. CCL source code through an Operator GUI will be compiled to the intermediate "byte-like" code modules, and downloaded to each robot. There, an interpreter will execute it employing the kΩ-optimization and using a native vehicle controller.

There are benefits to implementing the systems both as a compiler and an interpreter that will be examined. A benefit to implementing it as a compiler is the efficiency of compiling a mission; it will only be computing information once. The CCL interpreter will be able to give the vehicle(s) commands piece – wise with feedback from sensors passed to the interpreter real-time which may be better than a compiler for such a dynamic environment. The most important benefit is adding the intermediate level which makes our design scalable and portable.

The back end in a compiler is where the written language is turned into a computer code that can be understood by the operating system on the vehicle. In our case, this will be hardware independent "byte code" for the CCL Virtual Machine.

Optimization for the Solution of Real-World Hard Problems

We will assume optimization to cope with dynamic aspects of the environment, the CCL interpreter on the vehicle will adaptively look for the best sequence of elementary behaviors for the byte code it has received. A static plan of action will not work in most cases. Thus planning for the best actions will be done incrementally.

To do this, the interpreter will build trees of potential solutions and search for the optimal sequence of actions.

The basic $-calculus search method used for problem solving is called the kΩ-optimization. It is a very general search method, allowing the simulation of many other search algorithms, including A*, Minimax, dynamic programming, tabu search, or evolutionary algorithms.

The problem solving works iteratively; through the select, examine and execute phases. In the select phase the tree of possible solutions is generated up to k steps ahead, and agent identifies its alphabet (or behaviors) of interest for optimization in its set Ω. The agent may not always have a complete set of behavior due to limitations on vehicle configuration, controller or tasks. This means that the tree of solutions may be incomplete in width and depth (to deal with complexity). However, incomplete (missing) parts of the tree are modeled by silent $-expressions ε, and their cost estimated (i.e., not all information is lost). The above means that the kΩ-optimization can be complete and optimal if some conditions are satisfied.

In the examine phase the trees of possible solutions are pruned minimizing the cost of solutions, and in the execute phase up to n instructions are executed.
Moreover, because the $-calculus cost operator may capture not only the cost of solutions, but the cost of resources used to find a solution, we obtain a powerful tool to avoid methods that are too costly, i.e., the $-calculus directly minimizes search cost. This basic feature, inherited from anytime algorithms, is needed to tackle hard

optimization problems, and allows solving total optimization problems (the best quality solutions with the minimal search costs).

Controller Framework

ChaCha, a NUWC group control development tool, is an extension of techniques developed in [7], which is a behavior based control design using port-arbitrated behaviors (PAB). We have extended their approach to support real-time manipulations of behaviors and ports. There are wildcards for connection of behaviors both those currently running and those that will potentially run. Ports can be created and associated with a behavior during runtime. This will be a powerful capability in adapting to changing situations such as unanticipated sensor failures. The motivation for these extensions is to establish a highly dynamic capability for activation and organization of behaviors to support CCL commands.

An interpreter will need a wrapper so each controller already on the vehicle can understand the CCL. This will have to be a unique wrapper for each native vehicle controller. The CCL Interpreter will work through ChaCha to interface with the native vehicle controller. ChaCha is proposed as the framework for creating a vehicle wrapper. This interface may initially consist of creating a mission file in the format the native vehicle controller is used to receiving as input.

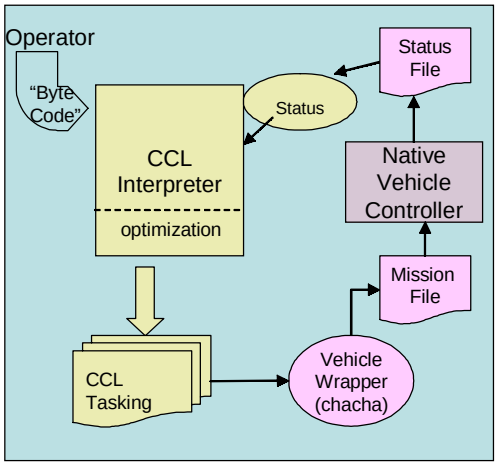

Figure 2: CCL interface with wrapper.

Surveillance Scenario

We have selected our demonstration mission to be a surveillance mission. It is intended to be generalized coverage of an area. The ability of a group to successfully survey an area has applications such as searching an area for targets, tracking a target, or simply as a forward protection to a high-value asset. The critical goal is to maximize coverage of an area while sampling. Critical tasks for the group are: maintaining a formation and assigning of sections of the area based on real-time own sensor footprint, energy/endurance and speed. We have defined a subset of atomic and complex behaviors that will support this mission. They are listed in Table 2.

Behavoir:	Parameters:
buildTeam	(no parameters, polls to see who is in the area)
getPos	longitude, latitude, object
collectSamp	instrument, areaSize, pattern
dataTransm	destNode, maxLength, ackReq, typeData, data

getPos: will get the longitude/latitude of the specified object

collectSamp: using a specified instrument will sample the given area in a given pattern (ex. lawnmower patter, circular pattern)

dataTransm: will send data to the given destination, specifying the data type and whether the communication is synchronous or asynchronous

Table 2: Subset of behaviors for Survey Mission

We represent a high level depiction of an example mission from the operator's point of view. Figure 4 starts with a role call of vehicles or assets in the area that are at the disposal of the operator for the mission. In our example we have used the WHOI REMUS platforms attempting to search an area. The operator defines a high level mission such as survey an area which is broadcast to the group as shown in Figure 5.

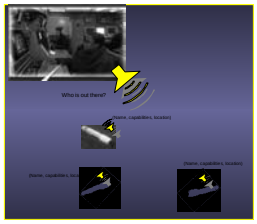

Figure 4: Role Call.

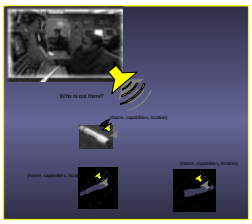

Figure 5: Send Mission

The group coordinates through their CCL modules to divide the area based on group size and ability as shown in Figure 6.

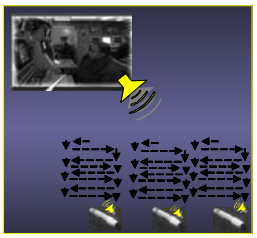

Figure 6: Coordinate.

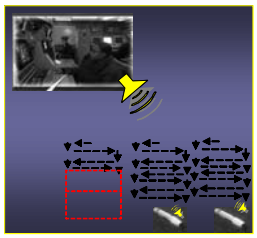

Figure 7. Recover.

In figure 7, one of the vehicles is unable to complete its mission (e.g., currents were high, energy source used up), so the remaining vehicles must coordinate through their CCL modules to assess the uncovered area and divide it between the two vehicles remaining in the group.

Summary

The language we are proposing will be a module that will interface through a "wrapper" for existing controller architectures to be used as a coordination tool. Different vehicles that must coordinate with each other will use the CCL wrapper to exchange information, commands, and acknowledgements. We are currently developing a CCL prototype for testing and evaluation.

Our long-term objectives consist of establishing a standard interface between groups of vehicles and supporting human operator to machine interaction.

References

[1] Duarte, C.N.,"The Distributed Control of Multiple Autonomous Vehicles for Mine Countermeasure for Littoral Operations", Undersea Defense Technology Europe 99 Conference, July 1-3, 1999, Nice, France.

[2] Duarte, C.N., Werger, B.B., "Defining a Common Control Language for Multiple Autonomous Vehicle Operations", Oceans 2000 Conference, Sept 11- 14, 2000, Providence, RI.

[3] Eberbach E., Brooks R., Phoha S., Flexible Optimization and Evolution of Underwater Autonomous Agents, in: (eds. N.Zhong, A.Skowron, S.Ohsuga) New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, Proc. of the 7th Intern. Workshop on Rough Sets, Fuzzy Sets, Data Mining and Granular-Soft Computing RSFDGrC'99, Yamaguchi, Japan, LNAI 1711, Springer-Verlag, 1999, 519-527.

[4] Eberbach, E., A Generic Tool for Distributed AI with Matching as Message Passing, Proc. Of the Ninth IEEE Intern. Conf. On Tools with AI ICTAI'97, Newport Beach, CA, 1997, pp.11-18

[5] Eberbach E., Enhancing Genetic Programming by $-calculus, Proc. of the Second Annual Genetic Programming Conference Gp-97, Morgan Kaufmann, 1997, 88 (a complete version in Proc. of the Tenth Australian Joint Conf. On AI AI'97, the ACS Nat. Committee on AI and ES, Perth, Australia, 1997, pp. 77-83).

[6] Eberbach, E., Expressing Evolutionary Computation, Genetic Programming, Artificial Life, Autonomous Agents and DNA-Based Computing in $-Calculus, Proc. Late-Breaking Papers of the Third Annual Genetic Programming Conf. GP-98, Univ. of Wisconsin, Madison, 1998, pp 33-41.

[7] B.B.Werger,"Ayllu: Distributed Port-Arbitrated Behavior-Based Control," Ullanta Performance Robotics Technical Report 98-01, To appear in Proceedings of DARS 2000.

[8] Brooks, R. Elephants Don't Play Chess, in Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back. (Ed.) Pattie Maes, MIT Press, pp3-15, 1993.

[9] Eberbach E., Phoha S., SAMON: Communication, Cooperation and Learning
of Mobile Autonomous Robotic Agents, Proc. of the 11th IEEE Intern. Conf. on Tools with Artificial Intelligence ICTAI'99, Chicago, IL, 1999, 229-236.

[10] Eberbach, E. A Proposal of the Generic Message-Passing Language for Interactive Automata Network of Autonomous Underwater Vehicles. Penn State, (working report), 1998a.

[11] Eberbach, E. An Interactive Automata Network Model and Its application to the Distributed Autonomous Unmanned Underwater Vehicles. Penn State, (working report), 1998b.

[12] Fedslund, J. and Mataric, M.J. Robots in Formation Using Local Information.

[13] Giampapa, J.A. and Sycara, K. Conversational Case-Based Planning for Agent Team Coordination.

[14] Howard, A., Mataric, M., Sukhatme, G. An Incremental Deployment Algorithm for Mobile Robot Teams. University of Southern California.

[15] Milner, R. *Communicating and mobile systems: the pi-calculus.* Cambridge University Press, 2001.