Toward a Generalized Architecture for Unmanned Underwater Vehicles

Changlong Lin, Xisheng Feng, Yiping Li, and Kaizhou Liu

Abstract—A common feature of unmanned vehicles is their complexity, which grows apace and provides its own challenges. Frameworks for managing this growing complexity have always been one of the key aspects of designing an unmanned vehicle. In this paper, a generalized architecture is proposed to not only address the complexity in developing an unmanned vehicle, but also support the algorithm exchange and technology transfer for integrating efforts from different researchers. We first detail the autonomous element, which is the fundamental unit of the architecture. Then the architecture is constructed, and thorough discussions are given. Finally, simulations on a semi-physical platform are carried out to examine the performance of this architecture.

I. INTRODUCTION

Unmanned vehicles have been widely used in the scientific, civilian, and military sectors. In order to function in unstructured, unknown, or dynamic environments, an unmanned vehicle should be able to perceive its surroundings and generate actions that are appropriate for the environment and the goals [1]. Unmanned Underwater Vehicles (UUVs) share common control problems with unmanned spacecraft, Unmanned Air Vehicles (UAVs), Unmanned Ground Vehicles (UGVs), and Unmanned Surface Vehicles (USVs), except for the limitation of communication in an underwater environment. Thus, research on UUVs is a part of the ongoing research efforts in the area of unmanned vehicles [2].

At the early stage, research on unmanned vehicles mainly aims at the validation of some basic functions. For this reason, the architecture is rather simple and explicitly visible. It can mostly work if we come to design of algorithms without any system structuring in the first place. However, as unmanned vehicles gradually come into use, they become much more complex to develop. It is no longer technically feasible to proceed with algorithms in a straightforward manner at the very beginning of system development. As a result, system structuring is inevitably paramount and also the only practical and effective starting point in the development of unmanned vehicles [3]. Hence, architectures form the backbone of complete robotic systems [4]; they are the frameworks where the following processes are implemented: control laws, errors detection and recovering, path planning, tasks planning and monitoring of the events along the execution of a particular mission [5]. The right choice of the architecture will greatly facilitate the specification, implementation and validation of robotic systems [4].

Since Shakey was presented in 1971, a spectrum of architectures have been developed and applied to different unmanned vehicles. Most of them can be classified into three categories: the deliberative architecture [6], [7], the reactive architecture [1], [8], and the hybrid architecture [9], [10].

II. TOWARD A GENERALIZED ARCHITECTURE

The design of an architecture is much more of an art than a science, thus it is really difficult to evaluate an architecture quantitatively. However, there are still several acknowledged qualitative criteria describing how a well-developed architecture should be [5], [11], [12], namely predictability, reactivity, robustness, modularity, extendibility, generality, and standardization. In our opinion, these criteria could be divided into two aspects. For the ability of unmanned vehicles, architectures should meet the needs of intelligent reasoning, dynamic reaction, and so on. For the development of unmanned vehicles, architectures should be generalized and standardized. Thanks to the rapid progress of computer engineering, algorithms which ran on the order of once every 10 minutes can now execute once every second. Therefore, the mismatches in planning and reaction times are no longer a compelling reason to enforce a strict separation of deliberation and reaction [13], thus the influence from the first aspect gradually fades away.

Manuscript received September 15, 2010. This work was supported in part by the National Basic Research Program of China under Grant No. 6138102008-4 and No. 6138101007-3, and the Key Project of Innovation Knowledge of Chinese Academy of Sciences under Grant No. YYYJ-0917. C. Lin, X. Feng, Y. Li, and K. Liu are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, 110016, China (phone: 86-24-23970212; fax: 86-24-23970076; e-mail: fly@ustc.edu, {fxs, lyp, kzhliu}@sia.cn).
C. Lin is also with the Graduate School of the Chinese Academy of Sciences, Beijing 100039, China.

978-1-61284-385-8/11/$26.00 ©2011 IEEE 2368
The rest of this paper is organized as follows. Section II describes the infrastructure of system’s fundamental unit, named Autonomic Element (AE) in details. Then the architecture is constructed and discussions are given in section III and section IV respectively. Section V demonstrates the simulation results on a semi-physical platform. Finally, conclusions are drawn in section VI.

II. DESIGN OF THE AUTONOMIC ELEMENT

Due to the great difficulties we have encountered in the development, maintenance, and upgrade of the control system, we initiate our work for a strategic resolution to tackle the complexity which induces these problems. Inspired by IBM’s autonomic computing paradigm [16], an autonomic element based architecture [17] is proposed and the complexity is well managed. However, as we proceed to refine it, we find that the autonomic element based architecture can be made generalized with some additional efforts.

Each independent part of the system is extracted into a node called autonomic element, which consists of three basic subassemblies: namely perception subassembly, decision subassembly, and database and knowledge subassembly, as shown in Fig. 1. Besides, it has six interfaces: reporter and receiver are used for interaction with its superior AEs, information exchanger and coordinator for peer AEs, and sensor and executor for its resources respectively. (The resources here refer to hardware devices if the AE is at the lowest level; otherwise they refer to adjacent subordinate AEs.) Human operator can also access any AE via information exchanger when debugging the system. Actually, these interfaces can be integrated into one by defining different types of messages. Here, we separate them for clarity.

A. Perception Subassembly

The perception subassembly collects the information from its resources via sensor, then processes, compares, and evaluates this information to obtain the best estimation of itself and the environment.

1) Information Collecting

The information which the perception subassembly collects can be classified into three categories:

--Data information. To low level AEs, data information refers to serial, parallel, and analog signals; whereas to high level AEs, it contains the position, attitude, and velocity of the vehicle, position and movement of the target, and so on.

--Status information. AE requires monitoring the status of its resources to assess its capabilities. For example, if the navigation AE detects the malfunction of the altimeter, it will delete the capability of altitude keeping from its capability list.

--Task progress information. Not all subordinate AEs provide this information. For a cruising task, it may be the percentage of completion of the course, the remaining distance, and so forth.

2) Data Fusion

After collecting the information, data processing is carried out to get the primary estimation, which will be compared with the prediction produced by the world model and knowledge subassembly. If the deviation is smaller than the threshold, the primary estimation and the prediction are synthesized to reason out the best estimation of the state and environment of the AE; otherwise, they will be passed to the fault detection module to check whether a fault occurs.

In addition, the perception subassembly needs to confirm the correctness of malfunction reports from its subordinate AEs according to its state. For instance, when the navigation AE receives a “no data” malfunction report from the GPS AE, it should take the depth of the UUV into consideration. If the UUV is working on the surface, the malfunction is proved; it will lose the capability of correcting its X-Y position. On the contrary, if the UUV is operating underwater, it is in the nature of things that the GPS AE fails to receive its signal; hence the navigation AE should neglect this report.

3) State Update and Malfunction Reporting

On one hand, the perception subassembly updates the world model and knowledge with its best estimation of the state and environment. On the other hand, when it detects a malfunction or its resource has just recovered from a fault, it will report to its superior AE that its AE loses or regains the corresponding capability (or capabilities). In other words, an AE should notify its superior AE of its currently possessed capabilities in real time.

B. Decision Subassembly

Based upon the goals received from its superior AE and the best estimation of the state and environment by the perception subassembly, the decision subassembly creates a series of instructions to achieve the goals. After synchronization and validation, these instructions are assigned to corresponding subordinate AEs for implementation. Generally, these instructions could be divided into three types:

1) Controlling Instruction
Most instructions belong to this type. At a high level, these instructions may be trajectory following; at a medium level, they may be altitude or velocity controlling; and at a low level, they may be effectors acting or devices switching.

2) Autonomous Behavior Enabling/Disabling Instruction

This type of instruction is used for enabling or disabling some of the autonomous behaviors of its resources. Normally, autonomous behaviors are more deliberative at high levels and more reactive at low levels. Hence, these instructions provide a mechanism for the vehicle to decide whether it should behave in a manner that is more deliberative or more reactive according to its applications.

Furthermore, this type of instruction is also useful for technology transfer between different unmanned vehicles. For example, almost every vehicle equips with a GPS device, thus a GPS AE is contained in the control system for the management of the GPS device. For a UAV or USV, after the GPS device is switched on, it should provide the position information periodically. Once the GPS AE cannot receive data from the GPS device, it should initiate a fault recovery handler because a fault has probably occurred. However, it is not the case because a UGV may work in a building or a UUV may operate underwater. Hence, the same GPS AE suits different unmanned vehicles as long as its superior navigation AE enables the autonomous recovery behavior for the no data fault when applied to a UAV or a USV, but disables it when applied to a UGV or a UUV.

3) Parameters Setting Instruction

These instructions are mainly used for setting the threshold of the autonomous behaviors. For instance, when a UUV is cruising in a shallow water environment, its navigation AE may autonomously switch its control mode from depth keeping to altitude keeping in order to avoid collision with the sea floor. The vehicle AE can set this switching threshold according to the UUV’s maneuverability and the other factors.

C. World Model and Knowledge Subassembly

The world model and knowledge subassembly provides the following functions.

1) Information Storing

The world model and knowledge subassembly stores all the information about its state and environment, including: the best estimation of its state and environment, the desired state, the current controlling instruction from its superior AE, the list of its capabilities, the status of its resources, and so on.

2) Process simulation

Combining the latest estimation of the state and environment and the instructions that would be sent to its subordinate AEs, this subassembly utilizes the world model to simulate and predict the state at the next time instant. Firstly, the result of the simulation is passed to the decision subassembly to validate the correctness of its instructions. Replanning (in the decision subassembly) is carried out to generate new instructions if previous ones fail to achieve a satisfying result. Secondly, the prediction is adopted as a reference for comparison in the perception subassembly to generate the best estimation. Thirdly, the deviation between the prediction and the sensory state (that is the primary estimation) is used to refine the world model.

3) Knowledge Management

Knowledge management is also offered to manage the algorithms and rules needed in the process of perception and decision making. Machine learning methods can be carried out to modify the knowledge.

D. Process Flow of AE

The process flow of the AE is shown in Fig. 2. After initialization, an AE goes into the idle state. When triggered by the arrival of a message, it acts accordingly: If the message is a query from its superior or peer AEs, it will reply with corresponding information. If the message is a signal from interior timer, it will collect the information from its resources and analyze it, then generate instructions and assign them to its resources. If the message is the status report from its subordinate AE (normally this indicates a malfunction), it will analyze whether its capabilities are affected, and decide whether a recovery process is required. If the message is an instruction from its superior AE, it will analyze and decompose the task into subtasks and assign them to its resources. When it finishes the procession of these messages mentioned above, it will return to idleness waiting for the next message. However, if the message is an ending signal, it will quit the loop and halt.

III. GENERALIZED ARCHITECTURE

Although all of the AEs share a common infrastructure, they are filled with different data, algorithms, and rules.
Hence they play different roles depending on where they reside in the architecture. It has to be noted that not all AEs accommodate every function described in section II; for instance, simulation is rarely required in the AEs for managing the devices.

In the architecture, every AE is capable of planning according to its current state and goal. In addition, it can respond quickly to the feedback and modify its actions. The AEs at the bottom level have the shortest spatial and temporal scope but with the most detail. Toward higher levels, the range in space and time increases, accompanied by the decrease in resolution. This limits the computational overload in any AE.

After we have developed all the AEs, we can obtain the whole control system. Via their “perception-decision” closed-loop, the AEs at the bottom level is responsible for the control and management of its corresponding devices; and then each higher level AE takes charge of several AEs at the adjacent lower level. Hence the system is constructed by a number of AEs integrated in a hierarchical and nested manner.

The generalized architecture of a UUV control system for the Intelligence, Surveillance, and Reconnaissance (ISR) mission is shown in Fig. 3. Generalized as it is, it can be easily adapted for another mission. Suppose that we would like to employ this UUV for the oceanography mission. Since the mission changes, the sonar should be substituted by a conductivity-temperature-depth (CTD) sensor. (We assume that this oceanography mission aims at collecting conductivity and temperature information.) Hence, the sonar AE is replaced by the CTD AE to manage the newly introduced sensor. And then the direct and indirect superior AEs that are associated with the sonar AE, namely detection AE, vehicle AE, and group AE, require modification due to the change of their subordinate AEs.

Firstly, it can be seen that most of the AEs remain unchanged when varying the mission. Secondly, as more and more systems under this architecture are developed, there will be a good collection of AEs whose management spans over all types of devices. Thus the process of substitution of an AE at the component level will be as easy as changing an accessory. Thirdly, each AE contains an augmented finite-state automata (FSA) which can be further divided into several sub-FSAs. In all these sub-FSAs, only a small part of them are mission specific. To those AEs that require modification, it is these mission specific sub-FSAs that should be redesigned. In a word, the generalized architecture is flexible for different missions, and reduces the work for the development, maintenance, and upgrade of the control system to the least.

IV. DISCUSSIONS

In this section, we discuss the generalized architecture from two aspects. One is associated with standards; the other is the comparison with related works.

A. Standards

A generalized architecture will not go too far without the support of a series of technical standards which include the following factors.

1) Terminology

Unified terminology provides a common language for unmanned vehicle community to communicate more effectively. Thus, the definition of terms is the first step toward standardization. National Institute of Standards and Technology (NIST) of USA has released its initial effort for unmanned systems [18], based upon which American Society for...
of Testing Materials (ASTM) refines them in the realm of UUV [19].

2) Interface

The generalized architecture also provides an open framework for cooperation: the AEs in the architecture may come from different laboratories. However, there are still two problems associated with the interface that need to be solved: message interface and instruction interface. Message interface details how AEs identify and communicate with each other. Instruction interface defines what an AE would receive from its superior and what it can assign to its subordinate; hence it specifies the duty and capability of the AE. There is a strong need for the unification of these interfaces so that AEs can not only seamlessly work with each other in an unmanned vehicle, but also be applied to different vehicles.

3) Technical Specifications

Some technical specifications will more or less affect the implementation of an unmanned vehicle and in consequence its control system and the architecture, for example, data format and payload interface. ASTM has developed several standards for UUV in these aspects [20]-[22].

B. Comparison

It is acknowledged that a generalized architecture is advisable for the integration of each other’s work; and some primary efforts have been made, for instance, the Joint Architecture for Unmanned Systems (JAUS) [23], [24], the All-Domain Execution and Planning Technology (ADEPT) architecture [25], and the 4-Dimension/Realtime Control System (4D/RCS) architecture [26]. We have borrowed a lot from these architectures during our work; however, there are still some differences between them.

Researchers have different opinions on what an intelligent closed-loop should be. For example, N. J. Nilsson uses “sense-plan-act”; R. W. Proud adopts “observe-orient-decide-act”; IBM researchers prefer “monitor-analyze-plan-execute”; H. Tianfield chooses “perception-decision”. We think that “perception-decision” would be more appropriate. For one thing, “sense”, “observe”, or “monitor” is practically the collecting of information from its resources; “act” or “execute” is actually the assignment of instructions. They are simple enough to be included in “perception” and “decision” respectively. Some may claim that “execute” also contains the process of supervising the implementation of instructions. But we argue that this process is actually another “perception-decision” closed-loop: an AE perceives the progress of the instructions and then decides whether to change it. For the other, it is widely acknowledged that the metrics for evaluating the autonomous capability of unmanned vehicles can be divided into three axes: situation awareness, decision-making, and external interaction [19]. Thus this choice is in accordance with the standard. For these reasons, an AE consists of perception, decision, and database and knowledge subassemblies.

In our architecture, we introduce not only controlling instructions, but also autonomous behavior enabling/disabling instruction and parameters setting instructions. This has provided a mechanism for researchers to decide whether their vehicle should be more deliberative or more reactive according to the mission and environment. Besides, this also makes AEs more flexible for different types of vehicles.

V. Simulations

To examine the performance of this architecture, we have developed a semi-physical platform, which is composed of the control system, the virtual environment system, and the visualization system. The control system controls the real UUV when mounted in the vehicle; the virtual environment system contains the model of the vehicle and environment to produce the signals for sensors according to the output of the control system; and the visualization system displays the motion of the vehicle throughout the mission.

In the semi-physical platform, we can choose different levels of noise for every sensor. Besides, many types of malfunctions can be introduced during the simulation, for example, no data fault of GPS, overload fault of thruster, and so on. The timing of this platform is identical with the real world. Although semi-physical simulations are not as practical as field experiments, they are much easier to be conducted; moreover, they can detect most of the timing and logical errors of the control system. Thus, it is quite suitable for the validation of an architecture.

Fig. 4 shows the screenshot of the visualization system at the beginning of a simulation. The main window displays the 3-dimension vision of UUV in the environment. The route in horizontal plane and the depth information are shown on the left and the bottom respectively. We develop a control system using the generalized architecture for ISR and oceanography missions. Both of the missions are completed successfully and the trajectories in X-Y plane for them are depicted in Fig. 5 and Fig. 6, respectively. We have also carried out many other experiments to examine the UUV’s response to different faults, and we find that the UUV can always act correctly.

![Screenshot of the visualization system](image-url)
Although it took us several months to develop the control system for the ISR mission, we spent less than 2 weeks adapting it for the oceanography mission. This reflects the flexibility of the generalized architecture. Besides, its performance is also examined via semi-physical simulations. It is expected that its advantage will be more apparent as more systems are developed under this paradigm.

Next step of this research is to develop the associated standards. However, it is inevitably a hard and time-consuming job which requires the collaboration of the whole community.

Besides, this architecture is suitable for all unmanned vehicles. Due to the constraint of our experiment condition, we mainly focus on UUVs. The differences between UUVs and the others unmanned vehicles during the development of control systems are also the problems we need to consider in future.

VI. CONCLUSION AND FUTURE WORK

Although it took us several months to develop the control system for the ISR mission, we spent less than 2 weeks adapting it for the oceanography mission. This reflects the flexibility of the generalized architecture. Besides, its performance is also examined via semi-physical simulations. It is expected that its advantage will be more apparent as more systems are developed under this paradigm.

Next step of this research is to develop the associated standards. However, it is inevitably a hard and time-consuming job which requires the collaboration of the whole community.

Besides, this architecture is suitable for all unmanned vehicles. Due to the constraint of our experiment condition, we mainly focus on UUVs. The differences between UUVs and the others unmanned vehicles during the development of control systems are also the problems we need to consider in future.

REFERENCES

[1] J. K. Rosenblatt, “DAMN: a distributed architecture for mobile navigation,” J. Exp. Theor. Artif. Intell., vol. 9, no. 2-3, April-September 1997.

[2] K-P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, and G. A. Demetriou, “Control architectures for autonomous underwater vehicles,” IEEE Control Syst. Mag., vol. 17, no. 6, pp. 48-64, December 1997.

[3] H. Tianfield, “Structuring of large-scale complex hybrid systems: From illustrative analysis toward modelization,” J. Intell. Rob. Syst. Theor. Appl., vol. 30, no. 2, pp. 179-208, February 2001.

[4] E. Coste-Maniere and R. Simmons, “Architecture, the backbone of robotic systems,” in 2000 Proc. IEEE Int. Conf. Rob. Autom., pp. 67-72.

[5] P. Riaño, J. Batlle, J. Amat, and G. N. Roberts, “Recent trends in control architectures for autonomous underwater vehicles,” Int. J. Syst. Sci., vol. 30, no. 9, pp. 1033-1056, September 1999.

[6] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, “A deliberative architecture for AUV control,” in 2008 Proc. IEEE Int. Conf. Rob. Autom., pp. 1049-1054.

[7] D. R. Yoeger, A. M. Bradley, and B. Walden, “System testing of the autonomous Benthic explorer,” in 1994 Proc. IARP 2nd Workshop on: Mobile Robots for Subsea Environments, pp. 159-170.

[8] J. G. Bellingham, T. R. Consi, R. M. Beaton, and W. Hall, “Keeping layered control simple,” in 1990 Proc. Symposium Autonomous Underwater Vehicle Technology, pp. 3-8.

[9] S. M. Smith, “An approach to intelligent distributed control for autonomous underwater vehicles,” in 1994 Proc. Symposium on Autonomous Underwater Vehicle Technology, pp. 105-111.

[10] J. Yuh, Underwater Robotic Vehicles: Design and control. Albuquerque, NM: TSI Press, 1995.

[11] A. Oreback and H. Christensen, “Evaluation of architectures for mobile robotics,” Auton. Robots., vol. 14, no.1, pp.33-49, January 2003.

[12] W. Li, Wengfeng, C. Henrikk, and O. Anders, “Architecture and its implementation for robots to navigate in unknown indoor environments,” Chin. J. Mech. Eng. Eng. Ed., vol. 18, no. 3, pp. 366-370, September 2005.

[13] R. Murphy, Introduction to AI robotics. MA: The MIT Press, 2000.

[14] Office of the Chief of Naval Operations Washington DC (November 9, 2004), The Navy Unmanned Undersea Vehicle (UUV) Master Plan [Online]. Available: http://www.navy.mil/navydata/technology/uuvmp.pdf

[15] John Westwood (2007), The AUV Gamechanger Report 2008-2017 [Online]. Available: http://www.dw-1.com/shop/shop-infopage.php?longref=68-6

[16] IBM (2005, June), An architectural blueprint for autonomic computing (2nd ed.) [Online]. Available: http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White %20Paper%20V7.pdf

[17] C. Lin, S. Ren, X. Feng, Y. Li, and J. Xu, “Autonomic element based architecture for unmanned underwater vehicles,” in OCEANS ’10 IEEE Sydney, pp. 1-5.

[18] H. Huang, Terminology for Specifying the Autonomy Levels for Unmanned Systems, Version 1.0, NIST Special Publication 1011, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2004.

[19] ASTM Standard F2541, “Standard Guide for Unmanned Undersea Vehicles (UUV) Autonomy and Control,” ASTM International, West Conshohocken, PA, 2006, DOI: 10.1520/F2541-06.

[20] ASTM Standard F2545, “Standard Guide for Unmanned Undersea Vehicle (UUV) Physical Payload Interface,” ASTM International, West Conshohocken, PA, 2007, DOI: 10.1520/F2545-07.

[21] ASTM Standard F2594, “Standard Guide for Unmanned Undersea Vehicle (UUV) Communications,” ASTM International, West Conshohocken, PA, 2007, DOI: 10.1520/F2594-07.

[22] ASTM Standard F2595, “Standard Guide for Unmanned Undersea Vehicle (UUV) Sensor Data Formats,” ASTM International, West Conshohocken, PA, 2007, DOI: 10.1520/F2595-07.

[23] S. Rowe (2008), An Introduction to the Joint Architecture for Unmanned Systems (JAUS) [Online]. Available: http://www.openskies.net/papers/07F-SIW-089%20Introduction%20to%20JAUS.pdf

[24] JAUS Standard (June 27, 2007), The joint architecture for unmanned systems, Reference Architecture Specification [Online]. Available: http://www.jointrobotics.com/documents/jaus/Final%20RA%20V3-3%20Part 1.doc

[25] M. Ricard and S. Kolitz, “The ADEPT Framework for Intelligent Autonomy,” presented at RTO AVT Course on Intelligent Systems for Aeronautics, Rhode-Saint-Genese, Belgium, 13-17 May 2002.

[26] I. S. Albus, “4D/RCS: A reference model architecture for intelligent unmanned ground vehicles,” in 2002 Proc. SPIE Int. Soc. Opt. Eng., pp. 303-310.