Autonomic Element Based Architecture for Unmanned Underwater Vehicles*

C. Lin[1, 2], S. Ren[1, 2], X. Feng[1], Y. Li[1, 2] and J. Xu[1, 2]
1. State Key Laboratory of Robotics, Shenyang Institute of Automation
Shenyang, Liaoning Province, China
2. School of Graduate, Chinese Academy of Sciences
Beijing, China

Abstract- **The architecture has always been one of the key aspects for designing an unmanned underwater vehicle. The architecture should serve as an aid, not a burden, in the integration of modules that have been developed independently, so it must not be overly restrictive. In this article, three types of architectures (deliberative, reactive, and hybrid architecture) are reviewed. Then the criteria for evaluating architectures are discussed. By borrowing the idea of autonomic computing, the autonomic element based architecture for unmanned underwater vehicles is constructed. Finally, simulations are carried out on a semi-physical platform to validate the feasibility of this architecture.**

I. INTRODUCTION

Unmanned vehicles are becoming widely used in the military and civilian sectors. These include unmanned spacecraft (US), unmanned air vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned underwater vehicles (UUVs) [1]. UUVs share common control problems with US, UAVs, and UGVs except for the limitation of communication in an underwater environment. Thus, Research in UUVs is a part of the ongoing research efforts in the area of unmanned vehicles [2].

An unmanned vehicle may be defined as "a vehicle with a sensorial system and an actuator system, managed by a control architecture, and able to undertake a user-specified mission" [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 [3]. The right choice of the architecture will greatly facilitate the specification, implementation and validation of robotic systems [4].

In the past decades, a great number of architectures have been developed and applied to different unmanned vehicles. Most of them can be classified into three categories: the deliberative architecture, the reactive architecture, and the hybrid architecture.

A. The Deliberative Architecture

The deliberative architecture, also called the hierarchical architecture, adopts a top-down approach [5]. It uses a functional and hierarchical decomposition to divide the system into levels; the higher levels are responsible for the overall mission goals, while the lower levels for solving particular problems to accomplish the mission [6, 7]. It represents a well-defined tightly coupled structure and has the ability to reason and make predictions; however, it lacks the flexibility to react to dynamic environments. Examples of this architecture include the Autonomous Benthic Explorer (ABE) [8], the Experimental Autonomous Vehicle (EAVE) III [9], the Marine Utility Vehicle System (MARIUS) [10], and the Ocean Technology Testbed for Engineering Research (OTTER) [11].

B. The Reactive Architecture

On the contrary, the reactive architecture, also referred to as the behavioral architecture, follows a bottom-up approach. It consists of behaviors which run in parallel without a high-level supervisor. The behaviors continuously react to the situation sensed by the perception system and the vehicle's global behavior emerges from the combination of the elemental active behaviors [3]. Although it is well suited for dealing with highly dynamic environments, it fails to achieve non trivial objectives due to the lack of high-level control. In addition, as the number of behaviors increases, the synchronization between behaviors becomes more difficult and the system gets more complicated. The Odyssey II [12] and the Sea Squirt [13] are typical examples of this category.

C. The Hybrid Architecture

The hybrid architecture can be considered as a combination of the deliberative and the reactive architectures for taking their advantages and minimizing their limitations. It is widely recognized as the most suitable architecture for an unmanned vehicle. Examples of this architecture include the Ocean Voyager II [14], the Omni-Directional Intelligent Navigator (ODIN) [15], and the Phoenix [16].

Given a fairly general agreement on basic architectural principles, one might expect that a common software basis is available and that such a system is used for exchange of

* This work is supported by the 863 Program of China under Grant 2006AA04Z262, the Innovation Knowledge of Chinese Academy of Science under Grant No.07A6210601, the National Basic Research Program (973) of China under Grant No.6138101004-3, and the Key Project of Innovation Knowledge of Chinese Academy of Science under Grant No.YYYJ-0917.

978-1-4244-5222-4/10/$26.00 ©2010 IEEE

algorithms across laboratories and for technology transfer. However, this is unfortunately not the case [17]. To cope with this problem, this paper presents the autonomic element based architecture for UUVs which is the first step toward a single common architecture.

The rest of this paper is organized as follows: Firstly, the criteria for evaluating architectures and the drawbacks of the common hybrid architecture are presented. A brief introduction to the autonomic computing is given subsequently. And then, based on the idea of autonomic computing, the autonomic element based architecture for UUVs is constructed. Finally, simulations are carried out and conclusions are drawn.

II. Analyses of Architecture

There are several acknowledged criteria describing how a well-developed architecture should be. Oreback A. [17] briefly summarizes these basic requirements as:

1) Robot hardware abstraction. The portable architecture should provide abstraction of hardware such as sensors and actuators.

2) Extendibility and scalability. The architecture should support the adding of new software modules as well as new hardware to a system.

3) Limited run-time overhead. The computing ability of the hardware should be taken into consideration.

4) Actuator control model. In behavior based systems, the output from the individual behaviors need to be fused in order to produce one crisp actuator command.

5) Software characteristics. The system must be robust and reliable and has to be prepared to handle unexpected situations.

6) Tools and methods. Various tools can be used when constructing a software architectural system.

7) Well documented. In order for an architecture to achieve success, that is used apart from locally, the documentation has to be rigorous.

In our opinion, these criteria could be divided into two aspects. For one thing, as according to the abilities of unmanned vehicles, architectures should meet the need of intelligent reasoning, dynamic reaction, and so on. For the other, as according to the development of unmanned vehicles, architectures should be generalized and standardized. Thanks to the rapid progress of computer engineering, the influence from the first aspect gradually fades away. However, as the missions for unmanned vehicles become more and more complicated, the second aspect has become more practical then ever before.

At present, hybrid architectures are most popular and they are normally constructed in three layers. At the top, the planning layer transforms the mission into a set of tasks to be executed. The middle layer refines the tasks and controls their execution. At the bottom, the functional reactive layer consists of separate behaviors and performs repetitive calculations on raw or extracted data. Each layer contains several modules which are divided according to functions. Thus, an architecture contains many functional modules, for example, data collecting module, control module, errors detection and recovering module, path planning module, and so on. It is in accordance with human beings' intuition to organize in this manner, but this will lead to difficulties for the development and upgrade of the system. The reason for this is that the control and management of the vehicle are separated during the functional division. For example, the control and management of any device is distributed in data collecting module, servo control module, errors detection and recovering module, and so on; the control and management of navigation subsystem is achieved by path planning module, task planning and monitoring module, and others. Assume that we are going to employ a developed vehicle for a new mission. Since mission changes, the sensors and effectors equipped by the vehicle may change; and in consequence, all the modules mentioned above would require modification according to the change of devices. For this reason, this manner is quite inefficient for developing and upgrading a vehicle.

III. Autonomic Computing

"The advances in computing and communication technologies and software tools have resulted in an explosive growth in networked applications and information services that cover all aspects of our life. These services and applications are inherently complex, dynamic and heterogeneous. In a similar way, the underlying information infrastructure is large, complex, heterogeneous and dynamic, globally aggregating large numbers of independent computing and communication resources, data stores and sensor networks. The combination of the two results in application development, configuration and management complexities that break current computing paradigms, which are based on static behaviors, interactions and compositions of components and/or services [18]." As a result, today's computing and information infrastructures have reached a level of complexity that is far beyond the capacity of human supported system administration [19].

Inspired by the human body's autonomic nervous system, researchers in IBM proposed the notion of autonomic computing initiative in October 2001 to cope with this problem [20]. Via the implementation of autonomic managers, each of which automates some management function and externalizes this function according to the behavior defined by management interfaces, the system takes over most of the administration of itself and becomes self-managed. Self-managing is achieved by a control loop which consists of four fundamental parts, they are monitor, analyze, plan, and act. Self-managing mechanisms have a number of specific instantiations; IBM researchers have classified them into four categories as: self-configuring, self-healing, self-optimizing and self-protecting.

Autonomic computing paradigm has been put forward as a strategic resolution to tackling the complexity exponentially incurred in the system management since its launching. The realization of autonomic computing will result in a significant improvement in system management efficiency [19].

IV. Design of Architecture

From last section, we can see that the resources, including hardware devices and software applications, can only execute their instructions mechanically, they require manual administration to adapt to the dynamic environment. That is to say, they are automated but not autonomous due to the fact that the control and management of them are separated. Via the introduction of autonomic computing, the autonomic managers take over the administration from human operators thus the control and management of the resources are combined together and the resources become self-managed. Finally the problem of management complexity of information technology systems is well solved.

Now we turn back to unmanned vehicles. Unmanned vehicles are undoubtedly automated; in addition, they have to be autonomous because of the very limited intervention from operators during the missions. Errors detection and recovering module, task management module, and so on are typical examples which reflect that unmanned vehicles are self-managed. However, the combination of control and management is achieved only at the vehicle level. In our opinion, this is the fundamental reason why the common hybrid architecture becomes vulnerable for the development and upgrade of unmanned vehicles as missions gradually get complicated. As a result, we expect that the self-management characteristic is realized in every level and every granularity of the unmanned vehicle.

A. Autonomic Element

We extract every independent part of the system into a node called autonomic element (AE) whose structure is similar with an autonomic manager, and fill it with the control and management knowledge associated with it. Thus it becomes self-managed. As shown in Fig. 1, we replace the "monitor" and "analyze" in an autonomic manager with a single "perception", "plan" and "execute" with "decision", hence each AE consists of a perception subassembly, a decision subassembly, and a database and knowledge subassembly. The reason for this will be discussed later. In addition, each AE contains six interfaces to communicate with its superior, peer, and inferior AEs. Human operators can also interact with each AE via its "information exchanger" and "coordinator" interfaces.

In every control cycle, an AE will firstly obtain the data and status of its inferior AEs via "sensor". Then the information obtained will be fused in the "perception" subassembly and the result will be sent to its superior AE through "reporter" and stored in the "database and knowledge" subassembly. Subsequently, the "decision" subassembly synthesize the data and status in the "database and knowledge" subassembly, instructions from its superior AE via the "receiver", and coordination requests from its peer AEs to produce a series of optimized behaviors, which are finally sent to its inferior AEs via "executor". As a result, each AE contains a "perception-decision" close-loop for the control and management of its resources, which are hardware devices or inferior AEs. The process flow of the AE is shown in Fig. 2.

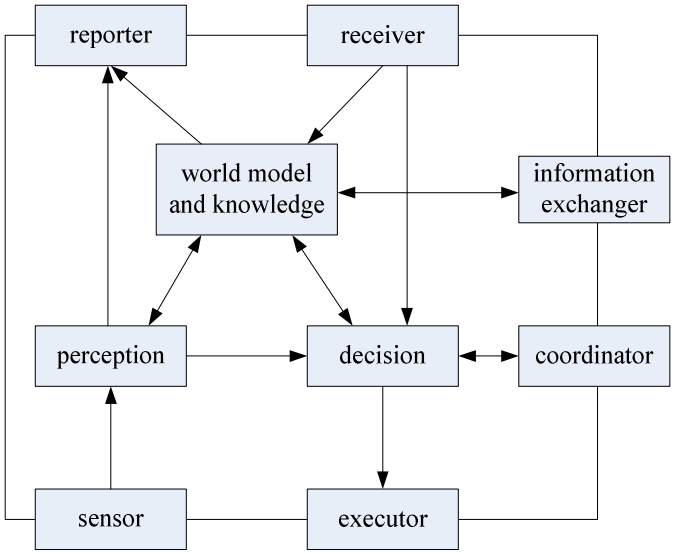

Figure 1. The autonomic element

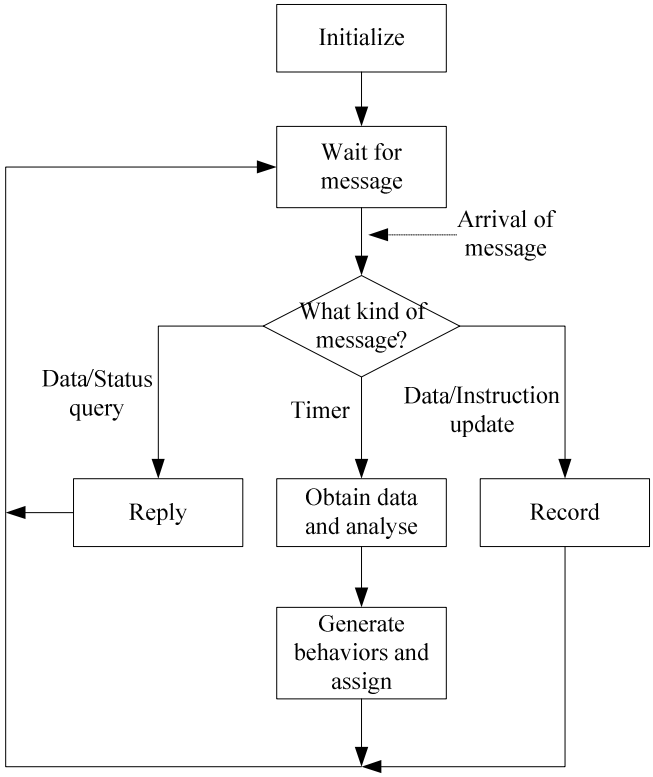

Figure 2. The process flow of the autonomic element

B. *Architecture*

After we have developed all of the AEs, we can obtain the whole control system by organizing the AEs according to their contents. Via its "perception-decision" close-loop, the AEs in the lowest level is responsible for the control and management of its corresponding devices; and then each higher level AE takes charge of several AEs in the adjacent lower level. Hence the system is constructed by a number of AEs combined in a hierarchical and nested manner, as shown in Fig. 3.

In higher levels, the control cycle is longer; the dimensions of space and time are larger; the data and status are more abstract; and the resolution of the instructions is lower. In the lower levels, the control cycle is shorter; the dimensions of space and time are smaller; the data and status are more detailed; and the resolution of the instructions is higher. Every AE is capable of planning according to its current status and goal; in addition, it can respond quickly to the feedback and modify its actions. As a result, the autonomic element based architecture is a hybrid architecture.

V. DISCUSSION

Since we expect to design a single common architecture that would support the exchange of algorithms across laboratories and for technology transfer, it is of great importance to have a common terminology definition and a serious of technical standards. The National Institute of Standards and Technology (NIST) of USA and other institutions have achieved some related efforts [21-25]. Based upon these, American Society of Testing Materials (ASTM) has drawn several standards for UUVs [26-29].

There are several methods to achieve or describe a close-loop. Nilsson uses "sense-plan-act" [30]; R. W. Proud adopts "observe-orient-decide-act" [31]; IBM researchers prefer "monitor-analyze-plan-execute"; H. Tianfield [32, 33] chooses "perception-decision"; and so on. We think that "perception-decision" would be more appropriate for two reasons. For one thing, "sense", "observe", or "monitor" is practically the process of obtaining data and status, hence it can be included in "perception". It is similar for "act" or "execute" to be contained in "decision". For the other, it is widely acknowledged that the metrics for evaluating the autonomous capability of unmanned vehicles can be divided in three axes, they are: (1) situation awareness, (2) decision-making, planning, and control, and (3) external interaction. Thus this choice is in accordance with the standard.

To the problem of dimension and resolution, 4D/RCS architecture advocates that range in space and time increase by about an order of magnitude, accompanied by an order of magnitude decrease in resolution at each successively higher level. However, based upon this, it seems that the number of levels of architecture may change with the duration and area of a mission. In our opinion, the duration and area of a mission will not affect the architecture, but the type of a mission. A three-level autonomic element based architecture is appropriate for single UUV missions and an additional level is expected for multi-UUVs applications.

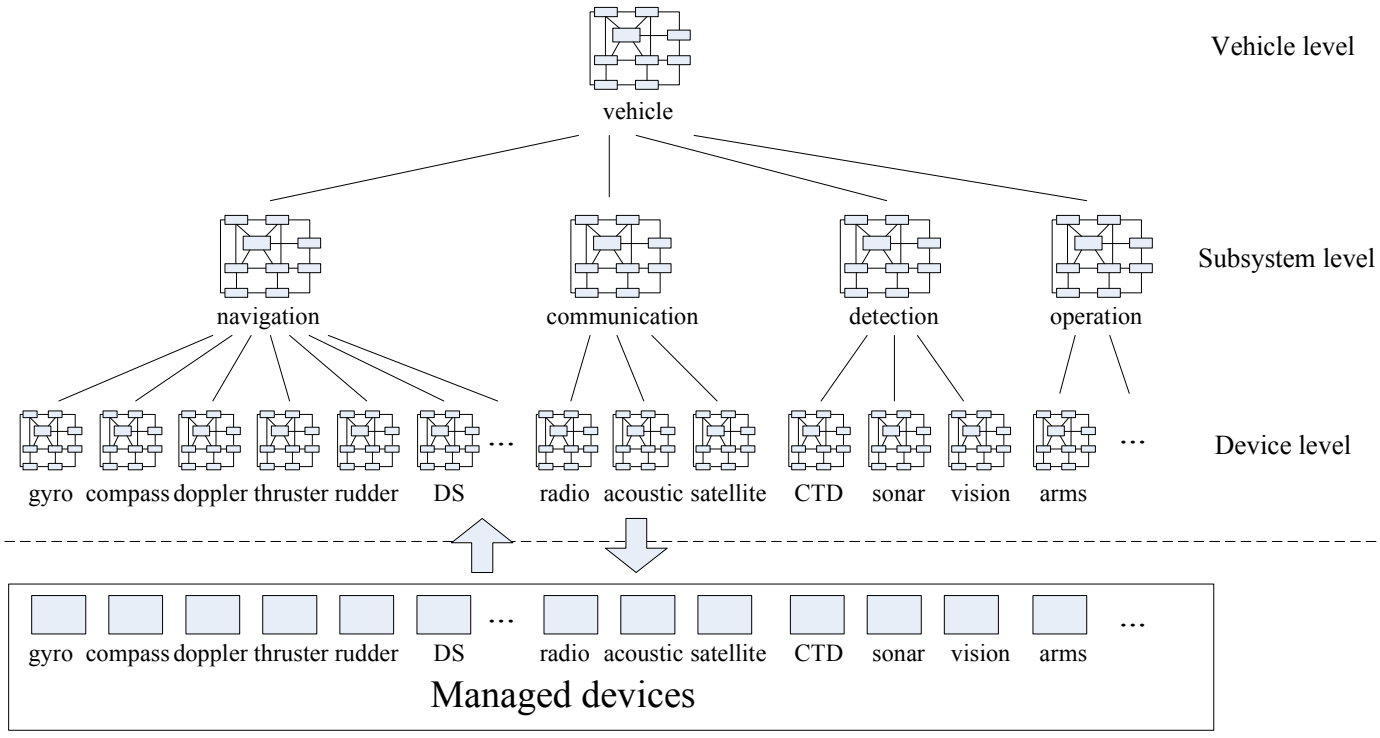

Figure 3. The autonomic element based architecture

VI. SIMULATION

We have developed an unmanned underwater vehicle control system using this architecture and simulations have been carried out on a semi-physical platform, as shown in Fig. 4. The main window displays the 3-dimension vision of UUV in the environment. The route in horizontal plane and the depth information are showed on the left and the bottom respectively.

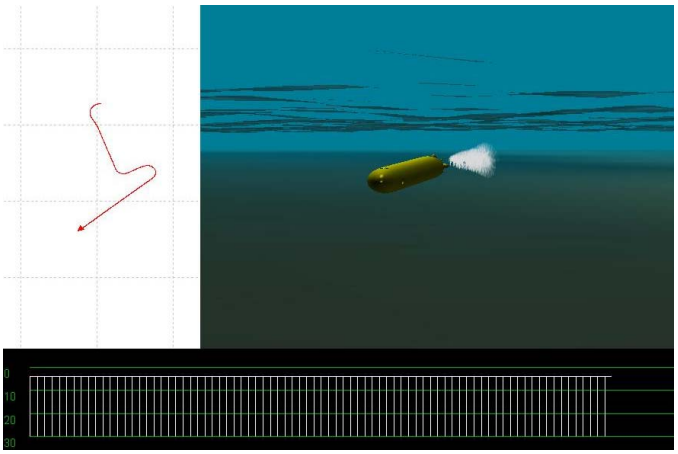

Figure 4. The simulation screen shot

VII. CONCLUSION

This paper has presented a new architecture for UUVs. Three types of architectures have been reviewed at first. Then the criteria for evaluating architectures and the drawbacks of the common hybrid architecture have been discussed. By borrowing the idea of autonomic computing, the autonomic element based architecture for UUVs has been constructed. Finally, simulations have been carried out on a semi-physical platform and the feasibility of this architecture has been validated.

REFERENCES

[1] L. N. Long, S. D. Hanford, O. Janrathitikarn, G. L. Sinsley, and J. A. Miller, "A review of intelligent systems software for autonomous vehicles," *Computational Intelligence in Security and Defense Applications, USA,* April 2007.

[2] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru and G.A. Demetriou, "Control architectures for autonomous underwater vehicles," *IEEE Control Systems Magn.*, vol. 17, pp. 48-64, 1997.

[3] P. Ridao, J. Batlle, J. Amat, and G. N. Roberts, "Recent trends in control rrchitectures for autonomous underwater vehicles," *Int. J. of Systems Science*, vol. 30, pp. 1033-1056, September 1999.

[4] E. Coste-Maniere, R. Simmons, "Architecture, the backbone of robotic systems," *Proc. of the IEEE Int. Conf. on Robotics and Automation, Canada*, pp. 67-72, April 2000.

[5] J. Albus, R. Lumia, and H. McCain, "Hierarchical control of intelligent machines applied to space station telerobots," *Trans. on Aerospace and Electronic Systems*, vol. 24, pp. 535-541, September 1988.

[6] E. Coste-Maniere, H. H. Wang, and A. Peuch, "Control architectures: What's going on?," *Proc. of the Int. Program Development in Undersea Robotics & Intelligent Control: A Joint U.S./Portugal Workshop, Portugal*, pp. 54-60, March 1995.

[7] K. Ganesan, S.M. Smith, K. White, and T. Flanigan, "A pragmatic software architecture for UUVs," *Proc. of the I996 Symp. on Autonomous Underwater Vehicle Technology Canada*, vol. 2, pp. 209-215, June 1996.

[8] D. R. Yoerger, A. M. Bradley, and B. Walden, "System testing of the autonomous benthic explorer," *Proc. of the IARP 2nd Workshop on: Mobile Robots for Subsea Environments*, pp. 159-170, May 1994.

[9] D. R. Blidberg et al., "The EAVE AUV program at the marine systems engineering laboratory," *Proc. of the IARP 1st Workshop on: Mobile Robots for Subsea Environments*, Canada, pp. 33-42, October 1990.

[10] A. Pascoal, C. Silvester, P. Oliveira, D. Fryxell, and V. Silve, "Undersea robotics research at IST: the AUV MARIUS programme," *Proc. of the Int. Program Development in Undersea Robotics & Intelligent Control: A Joint U.S./Portugal Workshop, Portugal*, pp. 111-118, March 1995.

[11] S. M. Rock, H. H. Wang, and M. J. Lee, "Task-directed precision control of the MBARI/stanford OTTER AUV," *Proc. of the Int. Program Development in Undersea Robotics & Intelligent Control: A Joint U.S./Portugal Workshop, Portugal*, pp. 131-138, March 1995.

[12] J. G. Bellingham and J. J. Leonard, "Task configuration with layered control," *Autonomous Underwater Vehicles Laboratory Report MITSG, The MIT Press, USA,* pp. 94-245, 1994.

[13] J. G. Bellingham and T. R. Consi, "State configured layered control," *Proc. of the IARP 1st Workshop on: Mobile robots for subsea environments, Canada*, pp. 75-80, October 1990.

[14] S. M. Smith, "An approach to intelligent distributed control for autonomous underwater vehicles," *Proc. of the I994 Symp. on Autonomous Underwater Vehicle Technology,* pp. 105-111, July 1994.

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

[16] A. J. Healey et al., "Coordinating the hovering behaviors of the NPS AUV II using onboard sonar servoing," *Proc. of the IARP 2nd Workshop on: Mobile Robotsfor Subsea Environments*, pp. 53-62, May 1994.

[17] A. Oreback and H. I. Christensen, "Evaluation of architectures for mobile robotics," *Autonomous Robots*, vol. 14, pp. 33-49, 2003.

[18] S. Hariri, et al., "The autonomic computing paradigm," *Cluster Computing*, vol. 9, pp. 5-17, 2006.

[19] H. Tianfield and R. Unland, "Towards autonomic computing systems," Engineering Applications of Artificial Intelligence, vol. 17, pp. 689-699, 2004.

[20] "An architectural blueprint for autonomic computing," *IBM Autonomic Computing White Paper*, 2006.

[21] H. Huang, E. Messina, and J. Albus, "Autonomy level specification for intelligent autonomous vehicles: interim progress report," *PerMIS Workshop, USA,* August 2003.

[22] H. Huang, et al., "Terminology for specifying the autonomy levels for unmanned systems, version 1.0," *NIST Special Publication 1011, USA,* January 2004.

[23] J. Albus, et al., "4D/RCS: A reference model architecture for unmanned vehicle systems, version 2.0," *NISTIR 6910, USA,* 2002.

[24] B. T. Clough, "Metrics, schmetrics! How the heck do you determine a UAV's autonomy anyway?" *Proc. of the Performance Metrics for Intelligent Systems Workshop, USA,* 2002.

[25] A. Meystal et al., "Measuring performance of systems with autonomy: metrics for intelligence of constructed systems," *NIST Workshop on Performance Metrics for Intelligent Systems,* August 2000.

[26] "Standard guide for unmanned undersea vehicles (UUV) autonomy and control," *ASTM USA,* 2007.

[27] "Standard guide for unmanned undersea vehicle (UUV) communications," *ASTM USA*, 2006.

[28] "Standard guide for unmanned undersea vehicle (UUV) sensor data formats," *ASTM USA*, 2006.

[29] "Standard guide for unmanned undersea vehicle (UUV) physical payload interface," *ASTM USA*, 2008.

[30] N. J. Nilsson, "SHAKEY the robot," *Technical Report Technical Note*, Artificial Intelligence Center, SRI Int., 1984.

[31] R. W. Proud, et al., "Methods for determining the level of autonomy to design into a human spaceflight vehicle: A function specific approach," *Proc. of the Performance Metrics for Intelligent Systems Workshop,USA,* September, 2003.

[32] H. Tianfield, "Structuring of large-scale complex hybrid systems: From illustrative analysis toward modelization," *J. of Intelligent and Robotic Systems: Theory and Applications,* vol. 30, pp. 179-208, February 2001.

[33] H. Tianfield, "An innovative tutorial on large complex systems," *Artificial Intelligence Review, An Intl. Science and Engineering J.,* vol. 17, pp. 141-165, April 2002.