Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

AUTONOMOUS UNDERWATER VEHICLE: PETRI NET BASED HYBRID CONTROL OF MISSION AND MOTION

ZONG-HU CHANG, XIN-QIAN BIAN, XIAO-CHENG SHI

Harbin Engineering University, Harbin, China
E-MAIL: changzonghu@sina.com

Abstract:

Terrain scanning is one of important missions of autonomous underwater vehicle. AUV's mission control covers a wide spectrum of research topics focusing on the interplay between event-driven and time-driven dynamical systems. The former is within the realm of discrete-event system theory, whereas the latter can be tackled using well-established theoretical tools from the field of continuous- and discrete-time dynamical system.

The paper describes a behavior arbitration hybrid software architecture of the mission control procedure which composes of mission level, task level and behavior level. The paper provides a mission control procedure model by adopting the formalism of extended Petri net theory. The task coordination algorithm based on the RW discrete event system theory has also been proposed to coordinate the pre-planned tasks and newly triggered tasks according to the inner or external events.

Details are given about the autonomous underwater vehicle at the present time, together with the simulation experimental validation of terrain scanning mission in the virtual system which has shown the reasonable hardware system, the implementation of the software and the correctness of the task coordination algorithm.

Keywords:

AUV; hybrid control; Petri net; discrete event system; task coordination

1. Introduction

It's a challenging job for the autonomous underwater vehicle to be sent on a terrain scanning mission in the unknown, unstructured sea environment. At first, the AUV receives the mission text from the surface workstation through net communication, the mission management computer plans the global path according to the electronic sea chart and the restrictions of the mission text, the task level plans the task sequences based on the global path and global rule databases. The task coordination module coordinates the task sequences according to clock events, hardware failure events and obstacle avoidance events. The task coordination may trigger the real-time mission replanning when it fails to coordinate tasks. In the worse condition, the AUV might give up the former mission and replan the task sequences to return to the callback point.

The task coordination module is also responsible for translating the task into vehicle primitives[1][2]. The behavior level arbitrates conflicts between vehicle primitives or conflicts between vehicle primitives and the sea environment, then outputs the vehicle primitives' parameters to the motion control computer. The motion control computer receives the vehicle primitives' parameters and real-time guidance information to calculate the motion control commands to drive the vehicle.

We can see from the mission control procedure that control system is a hybrid control system which combines the Discrete Event Systems (DES) and Dynamic Control of Continuous Systems (DCS). Software architecture is the key aspect which determines the intelligent level of the AUV. In the paper, we have defined a tri-level software architecture (a behavior arbitration hybrid software architecture) comprising mission level, task level and behavior level. The three levels separate the software into easily modularized functions encompassing everything from logically intense discrete state transitioning through the interfacing of asynchronous data updates with the real time synchronized controller functions that stabilize the vehicle motion to set points or trajectory commands.

2. Behavior arbitration hybrid software architecture

Behavior arbitration hybrid software architecture comprises three levels: mission level, task level and behavior and action level. Mission level is responsible for the mission planning and replanning; Task level coordinates the task sequences and outputs the vehicle primitives' parameters to the behavior and action level. Behavior level arbitrates conflicts between vehicle primitives and the conflicts between vehicle primitives and the sea environment, then outputs the fused commands to drive the vehicle to finish the mission.

0-7803-8403-2/04/$20.00 ©2004 IEEE

2.1. Mission level

The mission level with the highest intelligent mainly imitates the planning capability of human which reasons, plans the global path and forms the task sequences according to the global path and the restrictions of the mission text. When the task level fails to coordinate the pre-planned tasks, it will send task coordination failure message to the mission level and trigger mission replanning in real-time.

2.2. Task level

The task level can reason the environment, sensor system failures and reconfigure the system hardware resources. The level may trigger replanning messages to the mission level when it fails to coordinates tasks. The task coordination software is important to the mission's reliable and robustness. We define several kinds of tasks involving in the mission control procedure.

(a) Long range navigation task (T_navigating)
(b) Terrain scanning task (T_terrain_scanning)
(c) GSP calibrating task (T_GPS_calibrating)
(d) Obstacle avoidance task (T_obstacle_avoidance)
(e) Launching task (T_launching)

2.3. Behavior level

The behavior level is the lowest intelligent level among the three levels but with the highest precision. It organizes and arbitrates the vehicle primitives. A vehicle primitive is a parametrized specification of an elementary operation mode of an underwater vehicle, e.g. keeping a constant vehicle speed, maintaining a desired heading, holding a fixed altitude over seabed. This paper refers to the fuzzy logic algorithm to arbitrate the conflicts between the vehicle primitives.

As an example, in the long range navigation task, there exists the following vehicle primitives:

(a) Heading_keeping (VP_heading_keeping)
(b) Depth_keeping (VP_depth_keeping)
(c) Min_depth_keeping (VP_min_depth)
(d) Speed_keeping (VP_speed_keeping)

Each vehicle primitive includes several actions to drive the actuator control system.

3. Mission procedure model based Petri net

3.1 Mission procedure model

In order to finish the whole mission, the software system mainly includes the mission planning, task coordination, behavior arbitration modules and perception module. The main modules run in two computers: mission management computer and the motion control computer which form a distributing computer system. There exists several concurrent tasks in the two computers. The multi-process (multi-thread) mechanism can support the software system better, but multi-processes run concurrently causing the system more complex. In addition, the net communications between multi-processes enhance the complexity and import the randomicity more.

The design and analysis of mission procedure are built on the theory of Petri nets, which are naturally oriented towards the modeling and analysis of asynchronous discrete-event systems with concurrency. This approach leads naturally to a unifying framework for the analysis of the logical behavior of the discrete-event systems that occur at all levels of the mission control system. The paper introduces a procedure modeling concept and extended Petri net theory to model the mission control system. The sequence operator(;), or operator(||), and operator(&&), loop operator(*), parallelism operator([]), synchronous concurrent operator(=), synchronous blocking communication operator($), synchronous unblocking communication operator(&) have been defined to constitute the basic processes with the atomic processes, the basic processes form the whole procedure. We divided the mission control procedure into four modules which includes mission planning module (Plan&Replan), task coordination module (coor), motion control module (ctrl) and perception module (perc). The paper mainly discusses the mission planning module and task coordination module.

3.2 Mission planning module

Mission planning and replanning module is a procedure which runs in the mission management computer. The module includes three atomic processes: read the mission text, global path planning, task planning. The module is integrated by the three atomic processes using the sequence operator (;), synchronous blocking communication operator ($) which communicates with the task coordination module (coor). The procedure module can be formalized as:

Plan&Replan: Rnet; PathPlan; TaskPlan $ Coor

The module's Petri net description is illustrated in figure1, the meanings of the places and transitions refers to the reference[3]

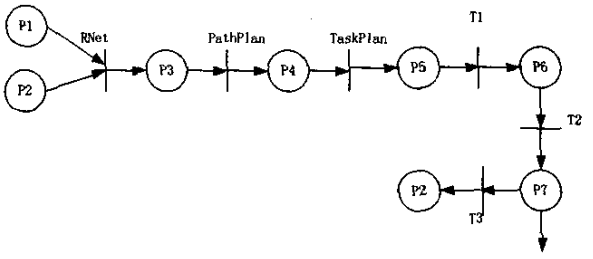

Figure1. Petri net description of the mission planning procedure

3.3 Task coordination module

Task coordination module (coor) includes three concurrent threads as receiving state thread, sending task commands thread and task coordination thread.

Receiving state thread (receive_state) is not only responsible for synchronous unblocking communication(&) with the motion control module, receiving the system states from the motion control computer (Ctrl), but also responsible for synchronous blocking communication($) with the task coordination thread (task_coor), receiving the states scheduling messages from the task coordination thread.

Sending task commands (send_task) thread is not only responsible for synchronous unblocking communication(&) with the motion control module, sending the task command messages to the motion control module, but also responsible for synchronous blocking communication($) with the task coordination thread, receiving the task scheduling messages as the current task types and task commands parameters.

The task coordination module can be formalized as:

Coor: (task_coor $receive_state& Ctrl)(=)(task_coor $ send_task& Ctrl)(=)(task_coor). The module's Petri net description is illustrated in figure 2.

In the task coordination thread, the main scheduling tasks include long range navigation tasks, terrain scanning tasks, GPS calibrating tasks and obstacle avoidance tasks. Place P9 can be decomposed as shown in figure 3, the meaning of places and transitions refers to the paper[3]

4. Real-time task coordination method based on R/W supervisory theory

4.1 Basic concepts of R/W theory

The R/W supervisory control is a pop subject of the automata and control realms since Y.C. Ho provided the concept of discrete event system(DES). It was emphasized by the experts and has produced a large amount of theories and methods such as mini-algebra, max-algebra, infinitesimal perturbation analysis and the discrete event supervisory control theory which was put forward by Wonham and Ramadge[4][5].

In the RW theory, the controlled object was described as an automata, the behaviors of the system were denoted as the strings (language) L. The states transitions were switched by the controllable events according to state supervisory controller.

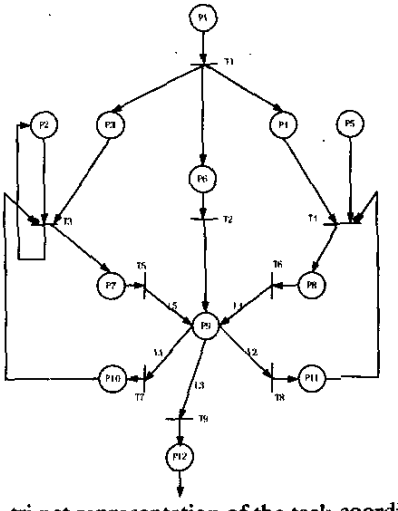

Figure2. Petri net representation of the task coordination module

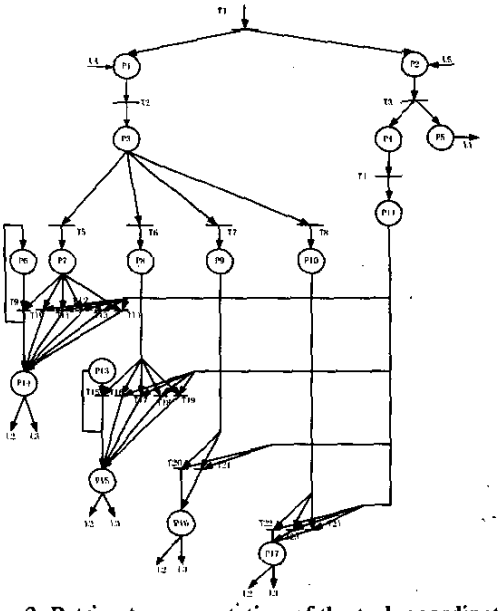

Figure3. Petri net representation of the task coordination thread

4.2 AUV's state supervisory controller

AUV's state supervisory controller supervises and assesses the task states according to the system information and acquires the event sequences.

The state supervisory controller has been expressed as the Petri net formulism:

$$\Sigma_o = (P, T, A, M_o) \quad (1)$$

$$\phi = P \rightarrow \Gamma \quad (2)$$

where:

Σ_o ——state supervisory controller, state transitions are triggered by the events in T

P ——places of supervisory controller

T ——fired events of the controller

$A: P \times T \cup T \times P$ ——flow relations between places and transitions

M_0 ——initial tokens

ϕ -----feedbacks of the outputs

Γ -----controllable events

We call that (Σ_o, ϕ) realizes supervisory controller f, if

$$\forall \sigma \in L_f, \ \phi(A(\sigma, M_0)) = f(\sigma) \quad (3)$$

Equation (3) shows if we trigger $f(\sigma)$ into the Σ_o, we can drive system Σ_o to some new state M_1. In our task coordination module, the state supervisory controller returns to its initial state M_0 when it triggers the highest priority event in each control circle.

The event sequences $\sigma_1, \sigma_2 ... \sigma_n$ are ranked according to their priorities. We define $MP(\sigma_i)$ to represent the priority of the event. $MP(\sigma_1) > MP(\sigma_2)$ means that the priority of the event σ_1 is higher than the event σ_2.

In every control cycle, we rank the event according to their priorities and get the event sequences as $MP(\sigma_1) > MP(\sigma_2) > ... > MP(\sigma_n)$. Only the event σ_1 with the highest priority can be triggered and the other events are prohibited in this control cycle, as shown in the equation (4).

$$f(\sigma_1) = 1, \ f(\sigma_2) = ... = f(\sigma_n) = 0 \quad (4)$$

The supervisory controller triggers the highest priority event σ_1, sends event message to the task coordination thread, then renews states to its initial way.

4.3 AUV's task coordination algorithm

As shown in figure 3, the Petri net formulism of the task coordination module can be described as:

$$G_0 = \{P, T, A, M_0\} \quad (5)$$

In equation (5):

$P = \{p_1, p_2, ..., p_n\}$ ---finite places, the places can be decomposed as

$P = P_{pre} \cup P_{pos} \cup P_{task} \cup P_{res} \cup P_o$, the subset of places that hold information related to the pre-conditions, resource allocation, task places.

$T = \{T_1, T_2, ..., T_n\}$ subset of the finite transitions

$A: P \times T \cup T \times P$ flow relations between the places and transitions

M_0 initial tokens of task coordination thread

The closed loop DES G_0 supervised by f can be expressed as (G_0, f), sometimes expressed as f/G_0. The basic problem in supervisory control is to modify the open loop behavior of DES G_0 to restrict its system states in the pre-defined ranges. The pre-defined ranges can be defined by giving the expected behaviors, or through some other performance guidelines. In this paper, The system behaviors are prescribed in a pre-defined manner to respond to the fired events.

As an example, if the supervisory controller supervises the heading error going beyond the limit event σ_1, obstacle avoidance event σ_2 in a long range navigation task, the supervisory controller produces the priority sequences σ_2, σ_1 as $MP(\sigma_1) < MP(\sigma_2)$. The supervisory controller triggers the obstacle avoidance event, as $f(\sigma_2) = 1$. In each control cycle, only the highest priority event can be triggered to ensure the real-time task scheduling. The supervisory controller makes up the obstacle avoidance message according to the obstacle avoidance events. The task coordination thread receives the message and pauses the current long range navigation task, schedules the obstacle avoidance task to run. The local path planning algorithm plans a safe plan to avoid the obstacle and finishes responding to the obstacle avoidance event.

5 Virtual simulation results of terrain scanning mission

A virtual simulation system[3] has been designed to test AUV's software architecture, the task coordination method and motion control method. The mission management computer receives the mission text that describes the mission type, the start point, the section of terrain scanning and the callback point. In addition, the mission text describes the dangerous areas and time restrictions to finish the mission.

The mission control system is currently implemented in hardware using two networked processors. All the mission level and task level software are written in Qnx watcom C&C++ language and run in a Gespac CompactPci computer with an Enthernet card communicating with the motion control computer.

The mission management computer receives the mission, makes global path planning and task planning to acquire the task sequences. In figure 4, we see that the task sequences include 5 long range navigation tasks, 5 terrain scanning tasks (in the CDEF area). There are no obstacles in the planned path. In order to test the obstacle avoidance algorithm and task coordination method, we add a sunken ship in the AB long range navigation task on purpose. The mission is created by linking the tasks together. Every task has two local pointers, previous and next, and four states, start, pause, stop and resume. When one task has been finished, the task's state becomes stopped and the task coordinator schedules the next task to run. At first, the mission management computer schedules the AB long range navigation task and sends vehicle primitives' parameters to the motion control computer. The vehicle travels under the commands of VP_heading_keeping, VP_min_depth and VP_speed _keeping in the autopilot control mode.

When the looking forward sonar finds the sunken ship as shown in figure 5, the task coordination module triggers the obstacle avoidance event, pauses the current long range navigation task and schedules the obstacle avoidance task. The obstacle avoidance task guides the vehicle to avoid the sunken ship using the local path planning algorithm. In figure 4, we can see the process clearly. When the vehicle steers clear of the obstacle, the module resumes the suspended long range navigation task to run.

From the depth curve chart shown in figure 6, we see that the inner clock has triggered 3 times GPS calibrating events in the BC long range navigation task. The alternation between the GPS calibrating tasks is about 600s.

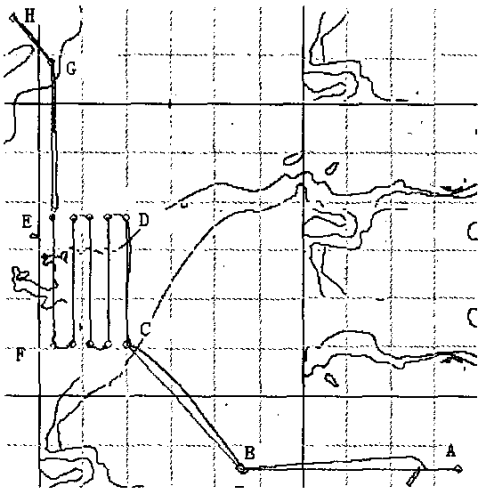

Figure4. AUV's terrain scanning mission control procedure

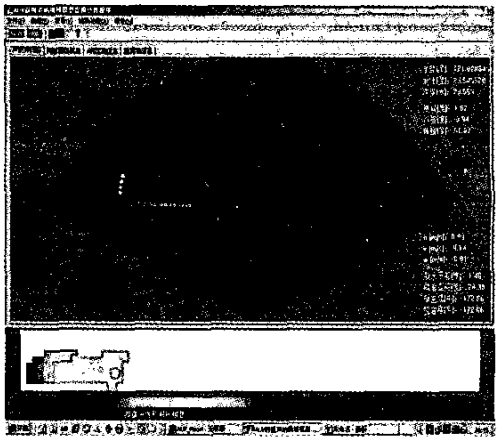

Figure5. Image of a sunken ship in the looking forward sonar

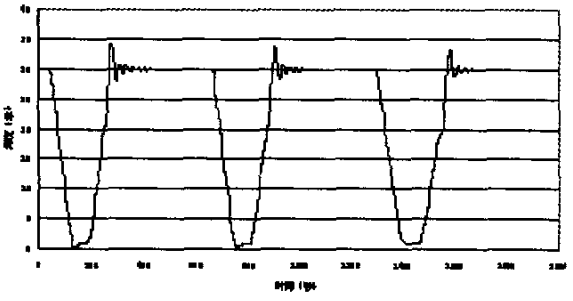

Figure 6. Depth curve of GPS calibrating task inserted in BC long range navigation task

6 Conclusions

The paper has built up the Petri net models of the mission planning, task coordination module and has provided the task coordination method based on the RW discrete event theory. At last, the simulation results of the terrain scanning mission control procedure in the virtual system have shown the reasonable hardware system, the implementation of the software and the correctness of the task coordination algorithm.

References

[1] D. Fryxell, P. Oliveira, A. Pascoal, C. Silvestre. Navigation, Guidance and Control systems of AUVs: An Application to the MARIUS Vehicle. IFAC Control Engineering Practice,March,1996.

[2] P.Oliveira, A.Pascoal, V.Silba, C.Silvestre. Design, Development, and Testing of a Mission Control System for the MARIUS AUV. International Journal of System Sciences, pp. 1065-1080

[3] Chang Zonghu, AUV software architecture and study of task coordination method, Ph.D. Paper, Harbin Engineering University, 2003,11

[4] Ho Y C. Performance evaluation and perturbation analysis of discrete event dynamic systems. IEEE Trans. Automat. Contr., 1987,32:563-572P.

[5] Ho Y C. and Cassandras.C.G. A New Approach for the Analysis of Discrete Event Dynamic System. Automatica, 1989,29(6):700-714