Research of Agent Based Control Model for Embedded Networked System

Haitao Zhang, Guiafng Wu, and Wenshao Bu

Abstract—**For the networked system composed of many embedded devices, the design of agent based control system may implement "Control on Demand". The existing modeling approach using formal languages has two shortcomings. Firstly it is difficult to get perfect formal model of agent based control. Secondly it is difficult to get real software agents. So we combine the advantages of UML and hierarchical coloured Petri Nets, and present a modeling methodology of agent based control for embedded networked system. After modeling static structure and dynamic behaviour of the system, we set up the mapping relation from UML model to Petri Nets model based on UML class diagram, collaboration diagram and state diagram, and get the corresponding Petri Nets model. The dynamic characteristic of the model is verified, the validated UML model is gotten, so as to get correct software code framework.**

Index Terms—**agent based control；networked system；embedded system；UML language；hierarchical colored Petri Nets**

I. Introduction

In recent years with the popularity of embedded processors and network technology, more and more devices embed processors and network chips. It can be predicted that in the near future most embedded devices will process the network function no mater whether the network function is needed.

In the networked system, a control algorithm is normally designed as part of the control system, and it is only valid in the system. Once it leaves the system, the control algorithm loses the effect and signification. With higher requirements of performance, the complexity of control algorithm and the cost of device will increase. For example, in order to make traffic control more intelligent, we must design the intelligent traffic intersection controllers and vehicular controller with higher cost. In addition, most embedded devices only have limited ability of computation and memory, and it is difficult for these devices to implement complex control algorithm, so traditional control theory including classical and modern control theory has great deficiencies. So some researchers introduce the agent technology into the network system, and put forward agent based control method [1-2]. The method transforms the design of control algorithm into the design of control agents, and different control algorithms are assembled by different control agents. Each networked device doesn't hold all control agents, but "control on demands", and only possesses the control agents which are currently needed. It can clearly be seen that the control method needs less memory and less computation ability, and embedded devices can attain higher performance and intelligence based on the principle of "local simplicity and remote complexity". The control method is fit for the running and management of the systems which are composed of numerous sensors and actuators that are distributed in different geography location, especially the systems such as traffic systems and vehicle systems.

The first step of designing agent based control system is to design system model so as to analyze and verify whether the system is correct, then we can get the source code framework of software agent from the model. The core of agent based control system modeling is the modeling of agents. Many researchers use different formal languages to model agents and get some achievements. There are the following methods of modeling multi-agents systems (MAS). (1) Luck and D'Inverno use Z language to describe the structure of agents in the abstract of different levels [3]. Although Z is precise formal language, it isn't fit to describe the complex conversation and interaction among agents, and is difficult to describe the dynamic property of agents; (2) Desire and Metatem use temporal logic to model agent based system [4]. However, the temporal logic utilizes complex logic including many math symbols to describe the application of MAS. It is very complex, and is difficult for the designer of MAS to develop systems; (3) Pi calculus is used to model MAS. Pi calculus can't express the information of physical structure of system, and concurrent behavior of system [5]; (4) All kinds of extended Petri Nets are used to model MAS [6-9].

All kinds of formal languages have its deficiencies. However, Petri Nets have strong dynamic analysis capabilities of the concurrency, asynchronization and uncertainty of the system. It is easily not only extended to satisfy different modeling requirements, but also has perfect math theory and simulation tool. Aiming the deficiency of classical Petri Nets, the hierarchical colored Petri Nets are applied to satisfy the requirements of agent based control system.

However, formal method is mainly used in analysis stage of agents, and cares for the conversation and intersection of agents. The formal method isn't fit to be applied to whole design and implementation stage of the system. In order to make up the deficiency of formal language, we introduce

Manuscript received August 1, 2018; revised 12, 16, 2018. This work was supported in part by the key scientific research project of universities and colleges of Henan Province under Grant 15A413003.

Haitao Zhang is with the School of Information Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China (corresponding author: e-mail: zhang_haitao@163.com).

Guifang Wu is with the School of Information Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China.

Wenshao Bu is with Electrical Engineering College, Henan University of Science and Technology, Luoyang, CO 471023 China.

UML language into the whole modeling of agent based control system so as to get perfect system model.

UML is very good at describing the static structure of the system, but it isn't fit to describe the dynamic behavior. However, Petri Nets have the stronger capability to describe dynamic behavior than static structure. UML model is lack of strict and effective verification and analysis method, but Petri Nets have perfect verification and analysis method, and is easy to simulate the dynamic behavior of systems. In addition, UML model is closely linked with program implementation [10]; According to their advantages and disadvantages, we combine the two modeling tools are to model the agent based control systems.

The rest of the paper is organized as follows. In Section 2, we introduce the whole modeling framework, and Section 3 presents detailed modeling method. Finally, a brief summary are discussed in Section 4.

II. Modeling Methodology

In the following, we first present the modeling framework combining the merits of UML and Petri Nets.

A. UML

UML is a graphical modeling language, and uses diagrams to describe static structures as well as dynamic behavior of systems. UML has becomes the standard in software engineering, and is growing in modeling of other domains.

The basic building block of UML is diagrams. UML uses a set of diagrams to reflect all aspects of a system, and each diagram shows a specific characteristic of the system. Each diagram is composed of a set of figures that include the important information on one aspect of the system. UML describes the static structure of the system by class diagrams and object diagrams, and describes the dynamic behaviors by state diagrams, collaboration diagrams, sequence diagrams and activity diagrams.

Of course, when we model a system, we don't need design all diagrams, but design the necessary diagrams according to real requirements.

Once UML model is designed by the software "rational rose", the source code framework is easy to be gotten from the software. The following UML diagrams are all designed in the software.

B. HCPN

As a modeling and analysis tool, Petri Nets are convenient to describe the concurrent and distributed system. If we only use classical Petri Nets to model and analyze the complicated systems, the model is very complicated and is difficult to be understood. So some extended Petri Nets appear including Stochastic Petri Nets (SPN), Colored Petri Nets (CPN), Object-oriented Petri Nets (OPN) and Hybrid Petri Nets (HPN). In multi-agent based systems, CPN is a valid analysis method. However, CPN model will become more complicated with more complicate model. So we use Hierarchical Colored Petri Nets (HCPN) to model agent based control systems. In the following, we first present the definition of HCPN.

Definition 1: Hierarchical Colored Petri nets are a 10-tuple HCPN=(S，SN，SA，PN，PT，PA，FS，FT，PP) where

S is a finite set of pages. , $\forall s \in S$, s is a non-hierarchical CPN. If $\forall s_1, s_2 \in S$ and $s_1 \neq s_2$, then they don't include common elements [11];

$SN \subset T$ is a set of substitution nodes;

SA is a page assignment function. It maps each substitution node to a page.

$PN \subseteq P$ is a set of port nodes.

PT is a function of port type. PT: PN→{in, out, i/o, general}.

PA is a function of port allocation that makes Socket nodes associate with port nodes.

$FS \subseteq PS$ is a limited set of fusion sets. For $\forall fs \in FS$, $p1 \in fs$ and $p2 \in fs$, it satisfies C (p1) =C (p2) and I (p1) =I (p2).

FT is a function of fusion type. FT:FS→{global, page, instance}；

$PP \in SMS$ is a set of prime page.

C. Modeling Methodology

In the following, we first present modeling framework combining the merits of UML and HCPN.

Shown as Fig.1, we first set up the system specification according the requirements of the system. Then the use case diagram of UML model is designed to capture the system requirements corresponding to system specification, and the class diagram is designed according to the existing objects and their functions. After that the collaboration diagram and state diagram are designed on the basis of class diagram.

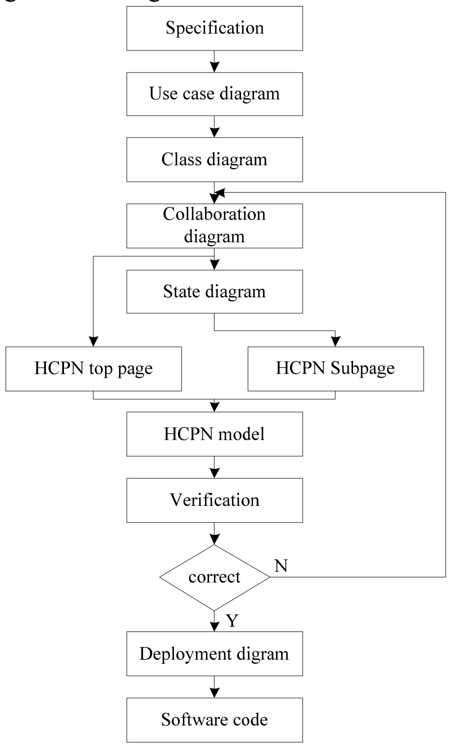


Fig.1. Modeling framework

After the collaboration diagram and state diagram are gotten, we map the two diagrams to HCPN model. Using the simulation tool and math theory we may get some characteristics of the model so as to decide whether the model satisfy the system requirements, such as deadlock, reachability, boundedness. According to the simulation result of HCPN model we may improve the corresponding UML model. Then we repeat the process of "map, verification and improve" until the verification results satisfy the system requirements.

This method integrates the merits of UML and Petri Nets. Petri Nets are used to do quantitative and qualitative analysis, then software code framework can be gotten from UML model.

III. SYSTEM MODELING

The core of agent based control method is the dynamic distribution of the algorithm. The control algorithm is not only changed at any time according to the requirement of the user, but also the device needn't change its hardware circuit. So the method increases the repeated utilization of source code, and implements the control on demand. The agent based control system has the following key parts: algorithm request, algorithm decomposition, control agent dispatch, and algorithm assembly. After the *algorithm decomposition agent* receives the request of a control algorithm form an embedded device, it decomposes the control algorithm into many basic control agents, and copy and dispatch these control agents utilizing the agent warehouse and other embedded devices. Then the *algorithm assembling agent* assembles all control agents, and implements the device control.

A. Use Case Diagram

It is necessary to construct the use case diagram to analyze and divide the system function when modeling systems using UML.

Fig.2 is the use case diagram of agent based control system. In Fig.2, the system user first releases the algorithm update command, and then the embedded device requests the design and decomposition to the *algorithm decomposer,* and requests the control agents to the region and center agent warehouse. Once the embedded device receives all control agents, it assembles them into the control algorithm.

B. Static Structure Diagrams

The allocation model of agent based algorithm mainly researches the allocation mechanism between the algorithm decomposing agent and assembling agent. In order to implement overall distribution in network, the system sets up the agent warehouse which is composed of large amount of basic control agents, and all control algorithms can be gotten by assembling these basic control agents. The agent warehouse copies and dispatches basic control agents according the request of embedded devices. Of course, if the nearby embedded devices possess the requested basic control agents, they can execute the task of agent warehouse.

The static structure mainly describes the relation among the embedded devices, the algorithm decomposer, the agent warehouse and control algorithm, and its UML class diagram is shown as Fig.3.

The algorithm decomposer and region agent warehouse is one-to-one relationship. The algorithm decomposer sends ID of agents which are dispatched by region agent warehouse and other embedded devices according to the agent information table. The region agent warehouse, the center agent warehouse, and embedded devices copy and dispatch basic control agents to the embedded device of requesting the algorithm. The dispatcher unit transmits the destination address and routing method to basic control agents by the function *ControlAgentDispatch*, and moves the basic control agents to the destination address.

In order to make agents move freely, the system must install the aglet platform, and uses the agent communication language ACL. In order to make the basic control agents reach the correct destination and are assembled into the requested control algorithm, the basic control agent need store routing line, algorithm number and assembling number. Its basic composition is shown as Fig.4 [12].

Agent tag	Byte number	Router table	Agent code

Fig.4 The basic composition of an agent

The migration of agents is implemented by the function *ControlAgentDispatch* which gives the next migrating decision according to the destination. After all basic control agents reach the embedded device, the device assembles all basic control agents and gets intelligent control algorithm agent. At this time the basic control agent has two roles which are basic control agent and intelligent control algorithm. In embedded network system, the basic control agent can copy and dispatch itself to other embedded devices according to the command of the algorithm decomposer; on the other hand, it is one part of intelligent control algorithm agent, and implements intelligent control of the embedded device.

C. Behavior Model

In order to research the dynamic behavior of agent based control system, the collaboration diagram and state machine diagram are constructed on the basic of UML class diagram.

Fig.5 is the collaboration diagram of an embedded device requesting new algorithm to the region agent warehouse through the network.

In order to get the state transformation relation of each object, in the following we give the state machine diagram of main objects. Fig.6-Fig.9 gives the state machine diagram of embedded device 1 requesting new algorithm to the region agent warehouse by the network.

Fig. 6 is the state machine diagram of embedded device 1, it uses the event *algorithm request*, *control agent receive*, *algorithm confirm* to model the algorithm request, receiving control agent, algorithm confirmation and the transformation of related states.

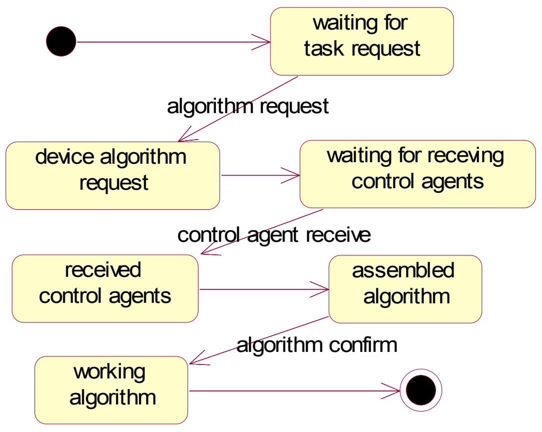


Fig.6 The state machine diagram of embedded device 1

Fig. 7 is the state machine diagram of the algorithm decomposer, it uses the event *algorithm design*, *algorithm decompose* and *receive region control agents* to model the state transformation relation of the algorithm request, new algorithm design, and receiving control agents.

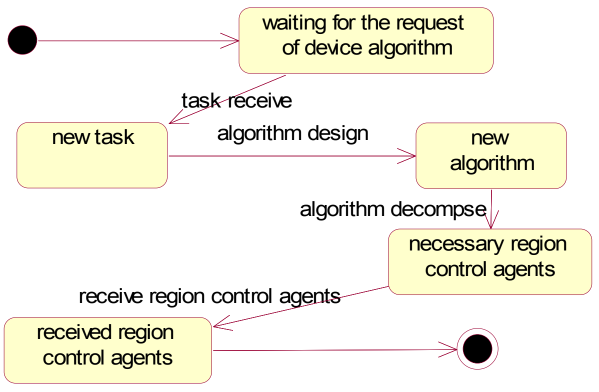

waiting for the request of device algorithm
task receive
new task
algorithm design
new algorithm
algorithm decompse
necessary region control agents
receive region control agents
received region control agents

Fig.7 The state machine diagram of algorithm decomposer

Fig. 8 is the state machine diagram of the region agent warehouse. Once entering into the state *region control agent request*, the region agent warehouse respectively sends the command of requesting agents to itself, other embedded devices, and center agent warehouse according to the distribution state of agents.

Fig. 9 is the state machine diagram of the intelligent control algorithm. It enters into the state *new algorithm ready* based on new assembled algorithm and device input.

Fig. 10 is the state machine diagram of embedded device 2, it uses the event *device control agent request*, *control agent copy*, *control agent dispatch* to model the request of control agents, copying control agent, dispatching control agents and the transformation of related states.

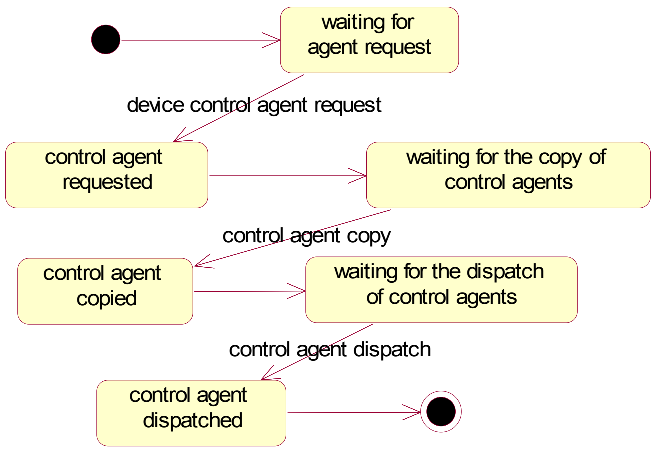

waiting for agent request
device control agent request
control agent requested
waiting for the copy of control agents
control agent copy
control agent copied
waiting for the dispatch of control agents
control agent dispatch
control agent dispatched

Fig.10 The state machine diagram of embedded device 2

In the system, the state machine diagrams of center agent warehouse are similar with that of embedded device 2, so we don't give its state machine diagram.

D. *Mapping from UML to HCPN*

It is easy to get software code framework from UML model, but it is difficult to verify whether the UML model is correct. So it is feasible method to introduce HCPN during the process of modeling systems using UML. Utilizing HCPN we may set up the method of verification and mathematical analysis of agent based control system so as to verify the validation of UML model, and lay the foundation for the design and implementation of embedded network system.

After designing the UML model, we should research the mapping method from UML to HCPN model, then verify the HCPN model using simulation tool "CPN Tools". The method avoids the deficiencies for the difficulty of getting the source code framework when we directly model the system using formal language.

UML model is based on the object-oriented idea, its class diagram includes all objects, and the collaboration diagram includes the action relation among objects. It can be seen that UML model shows receiving objects, sending message and internal operation of objects. In HCPN, the sub page reflects the internal action of the substitution transition, and the top page reflects the relation among sub pages.

In the following we give the steps by which HCPN model can be constructed from UML model.

1) Obtain the substitution transitions of the top page of HCPN model from UML class diagram.

A class represents a group of objects that have common state and behavior. We may realize the structural design of systems through the definition of class and descriptions of the relationships among the classes. Class diagram reflects the type of various objects type in the system, as well as the static relationships of the objects, so we may obtain the substitution transition of the top page of HCPN model from the class diagram.

The mapping rules from the class diagrams to the substitution transitions of HCPN are as follows [12]:

(1) In the class diagram the class that contains several operation functions is mapped to a substitution transition in HCPN model, and the name of the substitution transition may be same with the class name. In addition, the class that only contains one operation function is mapped to an ordinary transition with same name.

(2) The class that only has the static property in class diagrams is mapped to the place whose color sets is corresponding to the color sets of the class.

(3) Each operation in the class that is mapped to a substitution transition of the HCPN is replaced by a transition of the sub-page of this substitution transition.

In HCPN the direction of the arc is decided by the association direction of the class diagram.

2) Map the state diagrams to sub-pages.

State diagram captures the internal state transfer of user operations. An operation could be as small as a single class or as large as the entire system. State diagram is commonly used to model the transfer of an object. UML state diagram shows the different state of some objects and the state conversion relationships of the objects, and they describes the flow of the state of an object in its life cycle, and give the starting point and finishing point of state. State diagram displays the details of the dynamic behavior of objects, so it can easily be mapped to sub-page of HCPN.

According to the characteristics of the state diagram and the HCPN, a conversion event in UML state machine diagram is mapped to a transition of the Petri Nets, the source state is mapped into a pre-set place of the transition, the target state is mapped to a post-set place of the transition, and the conversion element (arc) is mapped to the two arcs which are from the pre-set place to the transition and from the transition to post-set place. The state diagram reflects states of the objects and external events which make states changed in UML, in HCPN the place and transition of sub-page reflects these relationships.

3) Map the collaboration diagram to the top page of HCPN model.

The collaboration diagram includes the transformation relation among objects. Utilizing the step 1), the top page is constructed from the collaboration diagram, and then the whole HCPN model of the system is gotten by the top page and the sub-pages.

E. *Construction of HCPN Model*

In Fig.3, there are five classes; in Fig.5, the class *embedded device* has two objects. According to the mapping rules of the UML to HCPN, HCPN model includes six substitution transitions: *embedded device 1*, *algorithm decomposer*, *region agent warehouse*, *center agent warehouse*, *control algorithm* and *embedded device 2*.

In UML model, the state machine diagram describes the state transfer of object under the event driving, and reflects the action during the lifetime period of the object. So the sub pages of HCPN model can be constructed by getting the information from the state machine diagram. Fig.11-Fig.16 is the sub pages of getting from the state machine diagrams of UML model according to the mapping rules.

In Fig.11, three transitions are respectively corresponding to three firing events of Fig.6. The place *device request tag* and *receive tag* model the firing sequence of three events.

In Fig.12, the variable *ags=[(1,"af"),(2,"be"),(3,"cg")]* of the transition *algorithm discompose* is to simplify the decomposition of requested algorithm. In the model the algorithm is supposed to be decomposed into three basic control agents that are respectively located in region agent warehouse, center agent warehouse and other embedded device.

In Fig.13, the variable *dags*, *ags2* and *ags1* of the transition *control agent region request* are respectively used to take out the basic control agents of region agent warehouse, center agent warehouse and other embedded device.

In Fig.14, the transition *algorithm assemble* is used to model the algorithm assembling, the transition *in* and *out* is used to model the device input and output.

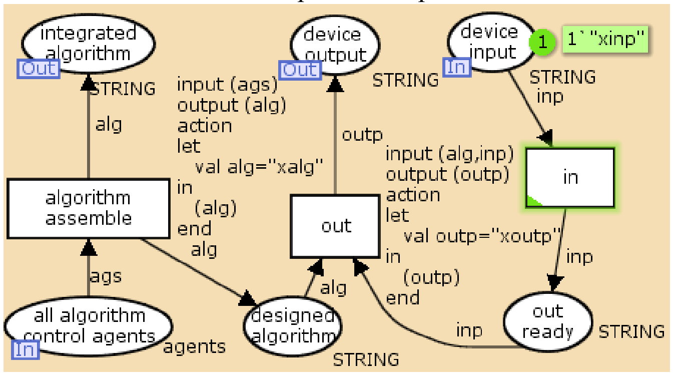


Fig.14 The HCPN sub-page of control algorithm

Fig.15 is the HCPN sub-page of embedded device 2.

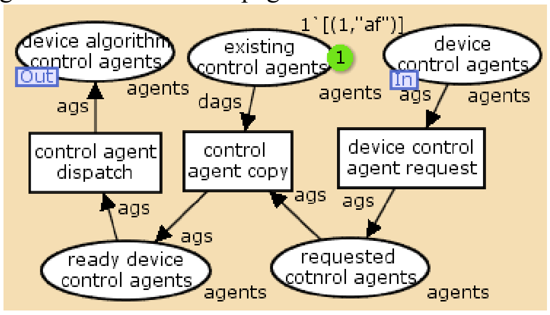


Fig.15 The HCPN sub-page of embedded device 2

Fig.16 uses the transition *center control agent copy* and *center control agent dispatch* to respectively model the process of copying and dispatching the agents after the center agent warehouse receives the request.

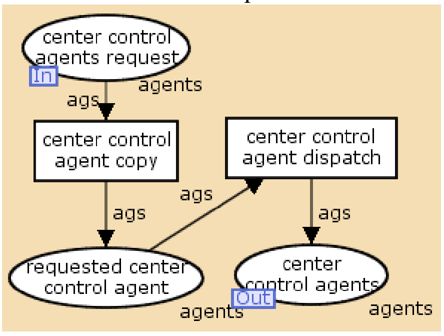

center control agents request
In
ags
agents
center control agent copy
center control agent dispatch
ags
ags
ags
requested center control agent
agents
center control agents
Out
agents

Fig.16 The HCPN sub-pages of center agent warehouse

The collaboration diagram reflects the information interaction, so we take sub page as the substitution transitions, connect all substitution transitions according to the collaboration diagram, and get the top page of HCPN model. Fig.17 is the top page of HCPN model.

F. *The Verification of HCPN Model*

Various analysis methods of Petri Nets can be used to do the strict qualitative analysis and validate the Petri Nets model that we get.

CPN Tool is the simulation software of HCPN model, and provides two analysis tools. One is to simulate the running of the model, and observe the changes of the mark of each place after a transition is fired. So we may get the properties of the model by single step or continuous running of transitions. The other is the State analysis tool which is called "Statespace". It can generate all states of HCPN model so that we can get all the properties by state graph [13]. In the following, we use the two analysis method to verify the HCPN model which is gotten from UML model.

Shown as Fig.18, it is the running result after executing 20 steps of transitions continuously. The place algorithm working gets a token "1" which represents that the vehicle has selected a route and is running.

Shown as Fig.19, it is the part results after running the analysis of calculating the state space and SCC graph. It represents that the state space and SCC graph have both 88 nodes and 178 arcs. The upper boundedness of the place algorithm working is "1" which represents that the vehicle has selected a route and is running. The dead and home marking nodes are both 88. Since the state space has only 88 nodes it must be the end node. In addition, the state space has not dead transition, live transition, and infinite occurrence sequence.

```
Statistics
------------------------------------------------------------------------
 State Space
   Nodes:  88
   Arcs:   178
   Secs:   0
   Status: Full
 Scc Graph
   Nodes:  88
   Arcs:   178
   Secs:   0

Boundedness Properties
------------------------------------------------------------------------
 Best Integer Bounds                                   Upper    Lower
   Top'algorithm_working 1                               1        0
   Top'all_algorithm_control_agents 1                    1        0
   Top'center_control_agents 1                           1        0
   Top'center_control_agents_request 1                   1        0
   Top'device_algorithm_control_agents 1                 1        0
   Top'device_algorithm_request 1                        1        0
   Top'device_control_agents 1                           1        0
   Top'device_input 1                                    1        0
   Top'device_output 1                                   1        0
   Top'integrated_algorithm 1                            1        0
   Top'region_algorithm_control_agents 1                 1        0
   Top'region_control_agents 1                           1        0
   Top'region_control_agents_request 1                   1        0
   Top'task_request 1                                    1        0
   algorithm_discompse_process'new_algorithm 1           1        0
   algorithm_discompse_process'new_task 1                1        0
   algorithm_discompse_process'task_receive_tag 1
                                                         1        0
   center_agent_warehouse_process'requested_center_control_agent 1
                                                         1        0
   control_algorithm_agent_process'designed_algorithm 1
                                                         1        0
   control_algorithm_agent_process'out_ready 1           1        0

     embedded_device1_process'device_request_tag 1       1        0
     embedded_device1_process'receive_tag 1              1        0
     embedded_device2_process'existing_control_agents 1
                                                         1        0
     embedded_device2_process'ready_device_control_agents 1
                                                         1        0
     embedded_device2_process'requested_cotnrol_agents 1
                                                         1        0
     region_agent_warehouse_process'agent_warehouse 1
                                                         1        0
     region_agent_warehouse_process'copied_agents 1
                                                         1        0
     region_agent_warehouse_process'migrated_agents 1
                                                         1        0
     region_agent_warehouse_process'requested_agent 1
                                                         1        0

       Best Upper Multi-set Bounds
         Top'algorithm_working 1      1`1
         Top'device_output 1          1`"xoutp"
       Best Lower Multi-set Bounds
         Top'algorithm_working 1      empty
         Top'device_output 1          empty

        Home Properties
       ------------------------------------------
         Home Markings    [88]

        Liveness Properties
       ------------------------------------------
         Dead Markings    [88]
         Dead Transition Instances    None
         Live Transition Instances    None

        Fairness Properties
       ------------------------------------------
           No infinite occurrence sequences.
```

Fig.19 The simulation results

Of course, if the simulation result doesn't satisfy the requirement, we can further to improve the UML and HCPN model until we get correct HCPN model.

From Fig.19, we know that the simulation result is correct by the above analysis. So we think that UML model is also correct. The correct source code framework may be gotten from UML model.

IV. Conclusion

For agent based control system composed of many embedded network devices, we have presented a methodology to support formal validation of UML model. The main idea is to map a HCPN model from UML diagrams so as to utilize the analysis techniques of HCPN. We construct key UML components of embedded network systems, and discuss the mapping activities: 1) Generation of sub-page of each class instance, 2) Generation of the top page of the model, and 3) Combine the top page and sub-pages to create a system-level model. The proposed methodology helps us to model and analyze the agent based control systems for embedded network devices.

References

[1] FeiYue Wang and ChengHong Wang. Agent-Based Control Systems for Operation and Management of Intelligent Network-Enabled Device. IEEE International Conference on Systems, Man and Cybernetics, US: IEEE Press, 2003, 5028 – 5033

[2] Kamil Hawdziejuk and Ewa Grabska. Cooperation of Agents in the Agent System Supporting Smart Home Control. International Conference on Cooperative Design, Visualization and Engineering, 2017, 57-64

[3] Michael Fisher. Representing and Executing Agent-Based Systems. Intelligent Agents – Proceedings of the International Workshop on Agent Theories, Architectures, and Languages. Lecture Notes in Computer Science, 1995, 890:307-323

[4] Yuanli Cai and Xinman Zhang. Formal Modeling Methodology for Multi-Agent Systems. Jounal of System Simulation, 2007, 19(14): 3151-3157

[5] Jianfeng Zhan. Research of Software Evolution and Dynamic nature under Internet Environment. Beijing: Institute of Software, Chinese Academy of Sciences，2002

[6] Dianxiang. Xu, Richard Volz, Thomas Ioerger, etc. Modeling and Verifying Multi-Agent Behaviors Using Predicate/Transition Nets. International Conference on Software Engineering and Knowledge Engineering 2002, 193-200.

[7] Bo Liu, Wei Li and Junzhou Luo. Semi-Online Scheduling Algorithm of Multi-Agent in Network Management. Journal of Computer Research and Development , 2006, 43(4): 571-578.

[8] YT Kotb, SS Beauchemin and JL Barron. Petri Net-Based Cooperation In Multi-Agent Systems. Fourth Canadian Conference on Computer and Robot Vision, 2007, 123-130

[9] Haitao Zhang and Shiwei Zhang. A method of algorithm update based on multi-agent for embedded networked systems. Advances in Information Sciences and Service Sciences, 2013, 5(9): 11-17

[10] Honghui Li, Aihua Zhao, Dalin Zhang and Junwen Zhang. Research on building software usage model based on UML model. International Journal of System Assurance Engineering and Management, 2018, 9(3): 675-683

[11] Bo Liu, Junzhou Luo and Aibo Song. Colored Petri Nets based Dynamic Multi-agent Scheduling Modeling and Analysis. Journal of System Simulation, 2007, 19(1): 193-198

[12] Haitao Zhang and Yanyan Li. Modeling and Analysis of Traffic Guidance Systems Based on Multi-Agent..Internatioanl Journal of Control and Automation, 2014, 7(2): 21-32.

[13] Étienne Andrél, Mohamed Mahdi Benmoussa and Christine Choppy. Formalising concurrent UML state machines using coloured Petri nets. Formal Aspects of Computing, 2016, 28(5): 805-845

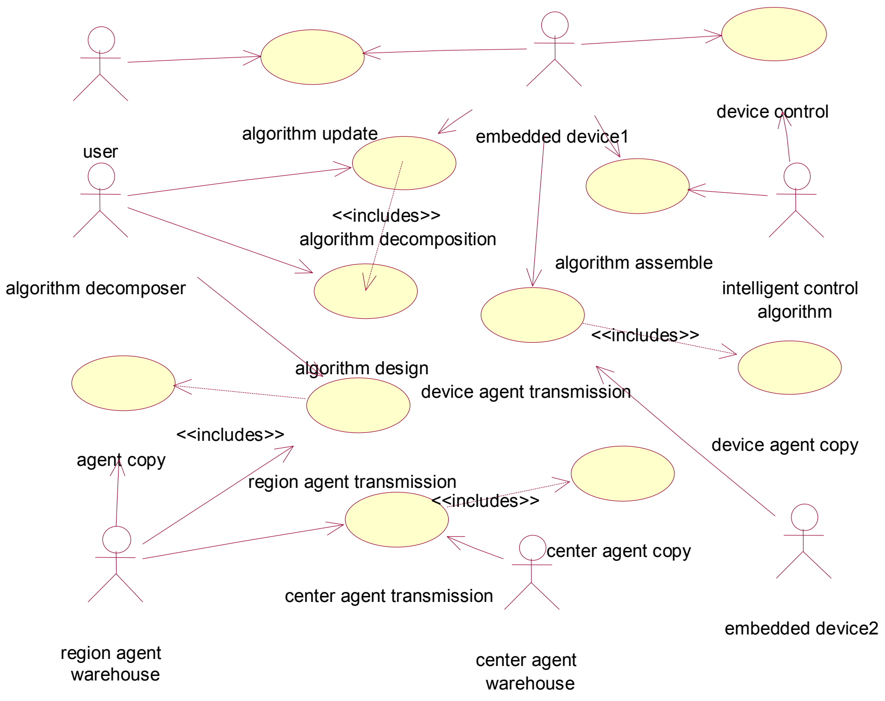


Fig.2 The use case diagram of agent based control system

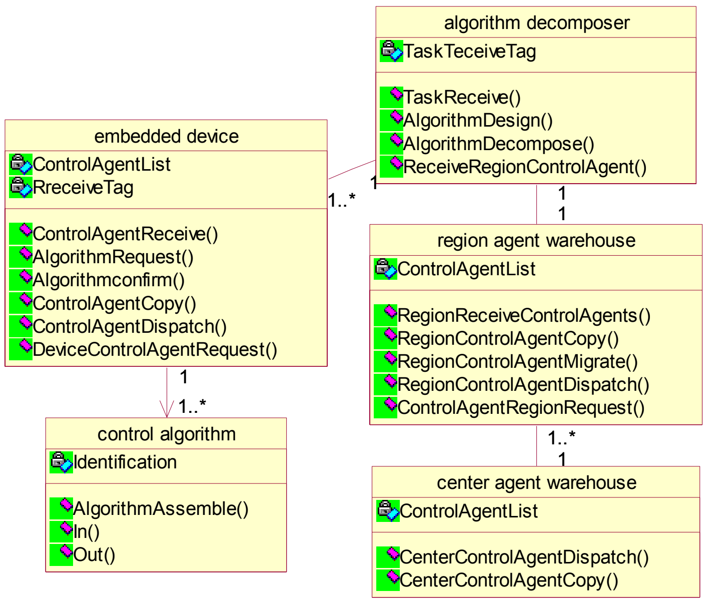


Fig.3 The class diagram of agent based control system

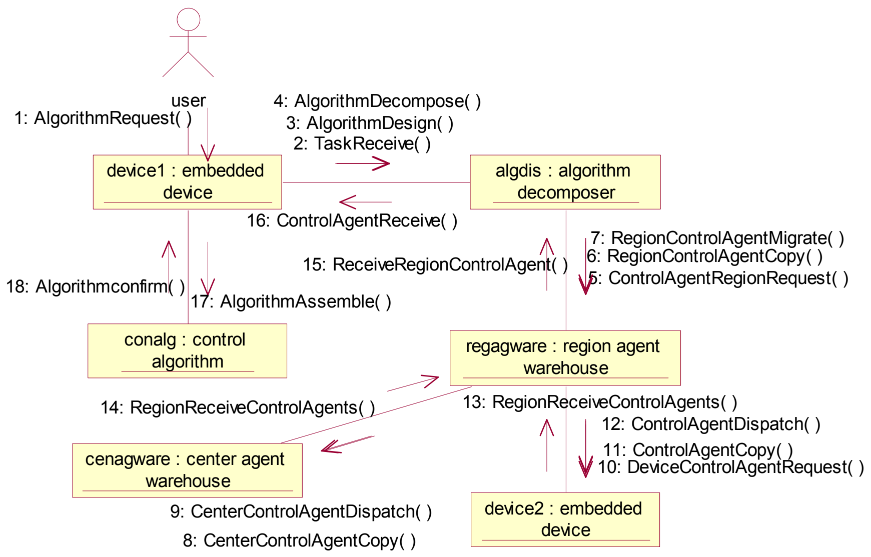

user
1: AlgorithmRequest( )
4: AlgorithmDecompose( )
3: AlgorithmDesign( )
2: TaskReceive( )
device1 : embedded device
algdis : algorithm decomposer
16: ControlAgentReceive( )
7: RegionControlAgentMigrate( )
6: RegionControlAgentCopy( )
5: ControlAgentRegionRequest( )
15: ReceiveRegionControlAgent( )
18: Algorithmconfirm( )
17: AlgorithmAssemble( )
conalg : control algorithm
regagware : region agent warehouse
14: RegionReceiveControlAgents( )
13: RegionReceiveControlAgents( )
12: ControlAgentDispatch( )
11: ControlAgentCopy( )
10: DeviceControlAgentRequest( )
cenagware : center agent warehouse
9: CenterControlAgentDispatch( )
8: CenterControlAgentCopy( )
device2 : embedded device

Fig.5 The collaboration diagram of algorithm requirement of embedded devices

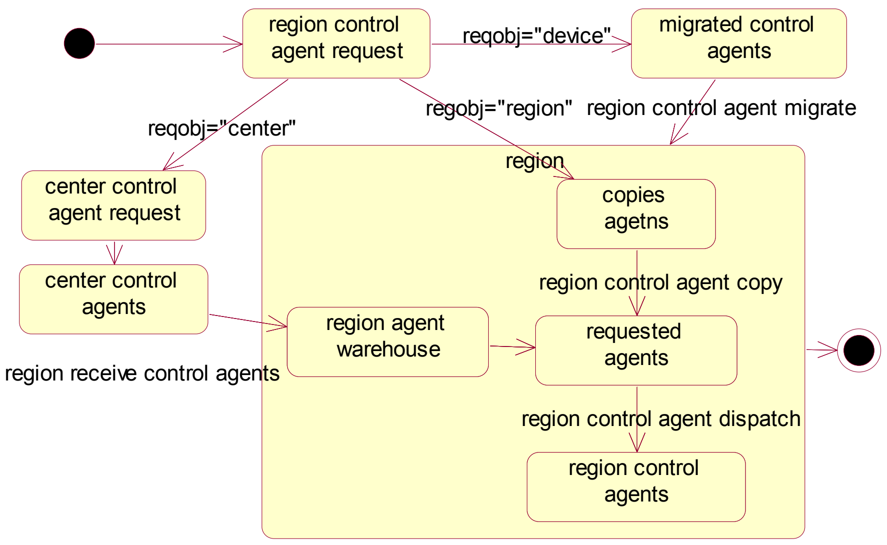

region control agent request
reqobj="device"
migrated control agents
reqobj="region"
region control agent migrate
reqobj="center"
region
center control agent request
copies agetns
center control agents
region control agent copy
region agent warehouse
requested agents
region receive control agents
region control agent dispatch
region control agents

Fig.8 The state machine diagram of region agent warehouse

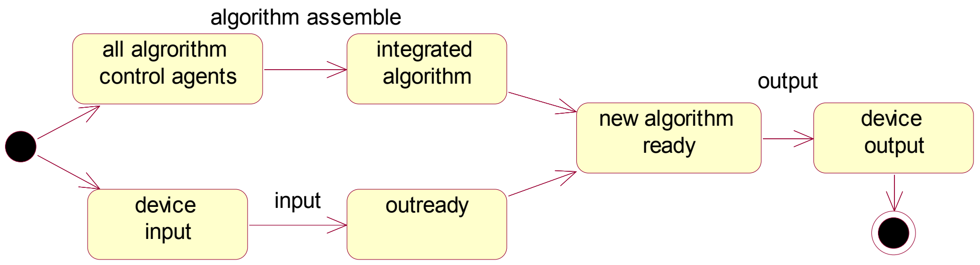
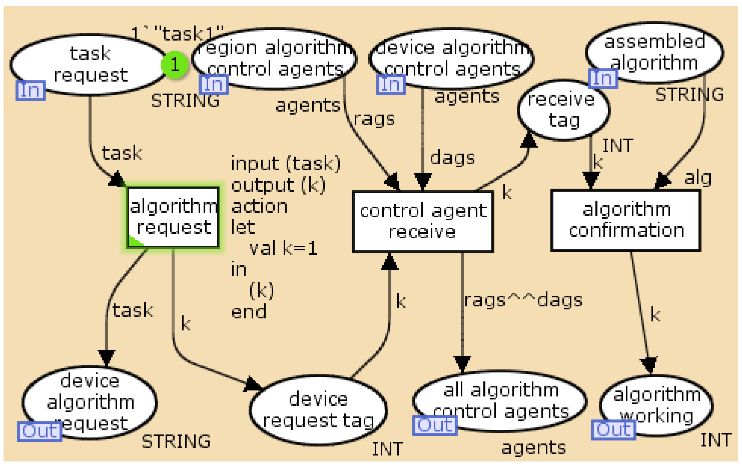
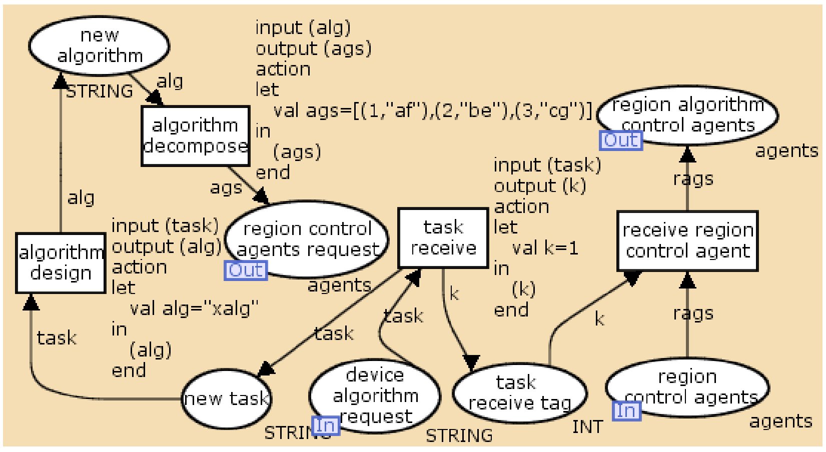


Fig.9 The state machine diagram of intelligent control algorithm



```
input (task)
output (k)
action
let
   val k=1
in
   (k)
end
```

Fig.11 The HCPN sub-page of embedded device 1

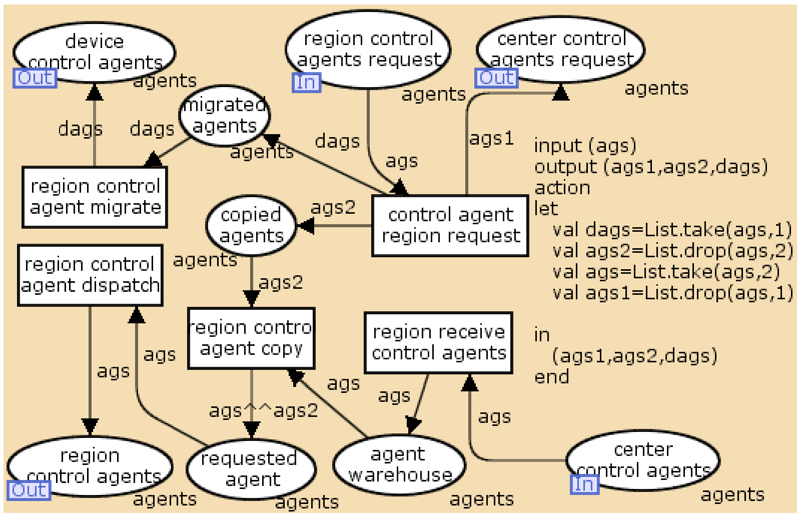


```
input (alg)
output (ags)
action
let
   val ags=[(1,"af"),(2,"be"),(3,"cg")]
in
   (ags)
end
```

```
input (task)
output (alg)
action
let
   val alg="xalg"
in
   (alg)
end
```

```
input (task)
output (k)
action
let
   val k=1
in
   (k)
end
```

Fig.12 The HCPN sub-page of algorithm decomposer


device control agents
Out
region control agents request
In
center control agents request
Out
migrated agents
region control agent migrate
copied agents
control agent region request
region control agent dispatch
region control agent copy
region receive control agents
region control agents
Out
requested agent
agent warehouse
center control agents
In
agents
dags
ags
ags1
ags2
ags^^ags2

```
input (ags)
output (ags1,ags2,dags)
action
let
  val dags=List.take(ags,1)
  val ags2=List.drop(ags,2)
  val ags=List.take(ags,2)
  val ags1=List.drop(ags,1)
in
  (ags1,ags2,dags)
end
```

Fig.13 The HCPN sub-page of region agent warehouse

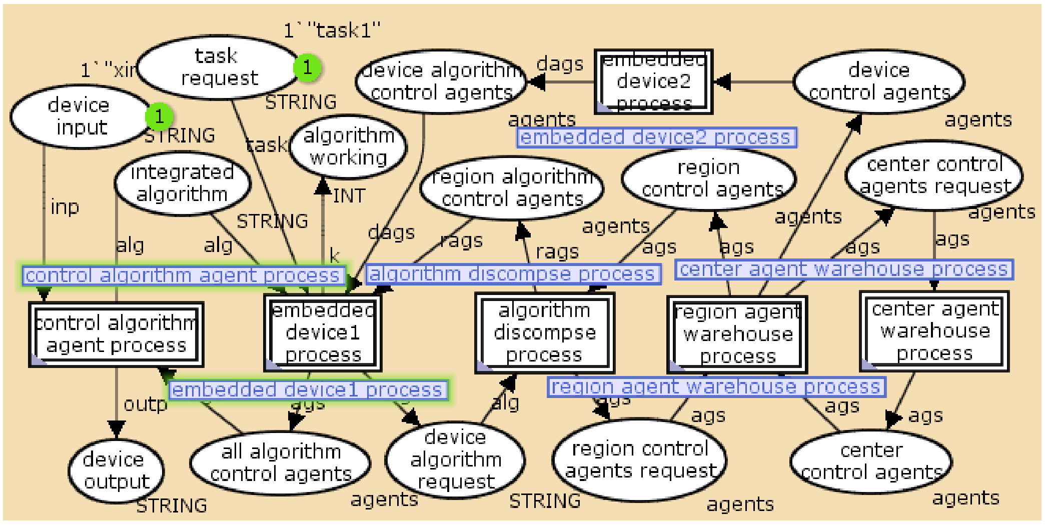

1`"task1"
1`"xin"
task request
device input
device algorithm control agents
embedded device2 process
device control agents
algorithm working
integrated algorithm
region algorithm control agents
region control agents
center control agents request
control algorithm agent process
embedded device1 process
algorithm discompse process
region agent warehouse process
center agent warehouse process
device output
all algorithm control agents
device algorithm request
region control agents request
center control agents
STRING
INT
agents
inp
alg
outp
ags
rags
dags
task
k

Fig.17 The HCPN top page of the system

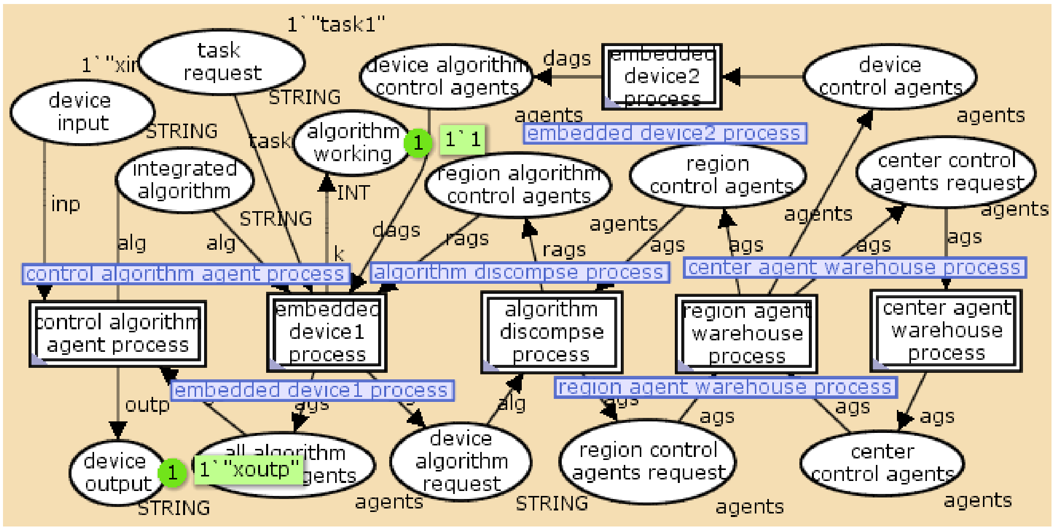

1`"task1"
1`"xin"
1`1
1`"xoutp"
task request
device input
device algorithm control agents
embedded device2 process
device control agents
algorithm working
integrated algorithm
region algorithm control agents
region control agents
center control agents request
control algorithm agent process
embedded device1 process
algorithm discompse process
region agent warehouse process
center agent warehouse process
device output
all algorithm control agents
device algorithm request
region control agents request
center control agents
STRING
INT
agents

Fig.18 The HCPN top page of the system after running 20 steps