Autonomic Element Design Based on Mind Agent Model

Maoguang Wang[1,2], *Jiewen Luo*[2], *Li Zeng*[2], *Zhongzhi Shi*[2]

[1]School of Computer Science, China University of Mining and Technology, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, China

Summary
In order to solve the increasing software and network management complexity, the idea of autonomic computing is brought forward by IBM. The aim is to increase the efficiency and decrease the management complexity, which achieves the goals of self-optimizing, self-healing, self-configuring and self-protecting. One of the critical questions is how to design an efficient and intelligent autonomic element for the autonomic computing system. Due to the intelligent agent characteristics, it is very suitable for agent to build autonomic element. Based on the agent research we build a mind agent model for the autonomic element, which has the autonomous and intelligent characteristics. Then we give a self-optimizing definition for the real application in AGrIP platform.

Key words:
Mind agent model, autonomic element, autonomic computing, self-optimization

Introduction

Due to the complexity of the network and distributed system applications, how to solve the software management and deployment complexity is becoming more and more concerned. In 2001 IBM releases a manifesto observing that the main obstacle to further progress in the IT industry is a looming software complexity crisis. Therefore IBM proposes the autonomic computing idea [1] to solve the software complexity crisis.

In accordance with autonomic computing idea, IT system should hold the capability to adjust itself without needing much manual intervention, namely, the system can execute autonomically and adapt itself to the changing environment. The autonomic computing inspiration comes from the autonomic nervous system, which can monitor the heartbeat, check the blood sugar and keep the body temperature without human interference. Autonomic computing includes four basic aspects: self-configuring, self-healing, self-optimizing and self-protecting[1].

To implement autonomic computing, the industry must take an evolutionary approach and deliver improvements to current systems that will provide significant self-managing value to customers without requiring them to completely replace their current IT environments [2].In the light of the autonomic evolution it is divided five levels: basic, managed, predictive, adaptive, autonomic. Basic level needs IT professional manual analysis and problem solving. Managed level takes advantage of centralized tools to collect distributed system information, thus decreasing the manual work time and cost. Predictive level correlates the system elements, predicts the optimal configurations and provides the guidance to the manager. Adaptive level can take actions according to the available resources and environment knowledge. The autonomic level can operate according to the dynamic business polices and the manager only needs to specify the policies and goals.

There are some key problems of autonomic computing which need to be solved successfully, such as the theory of problem determination, autonomic monitoring, complexity analysis, policy management and so on. They involve many disciplines including behaviour model, robust theory, coordination theory, autonomic statistical modelling etc. When designing the autonomic computing system, the prerequisite is to design the autonomic element. Now researchers are trying to make use of agent technology to construct autonomic computing systems [3, 4]. Intelligent agent has the characteristics of autonomous, self-learning, adjusting itself to the environment and so on. Therefore we focus on the application of agent technologies to the autonomic computing system.

The rest paper is organized as follows. Section 2 introduces the related work. Section 3 describes the mind agent model for autonomic element design. Section 4 illustrates the AGrIP platform architecture. Section 5 brings a summary of this paper.

2. Related Work

J.P Bigus, D.A.Schlosnagle, et al. [3] describe a toolkit for building multi-agent autonomic systems, which provides a

lightweight agent framework and a comprehensive library of intelligent software components and other test tools. But it just adopts the traditional simple agent structure though incorporating many relevant components for planning and decision-making.

Y.Diao, J.L.Hellerstein, et al. [4] propose an agent-based solution that not only automates the ongoing system tuning but also automatically designs an appropriate tuning mechanism for the target system. They study the problem of controlling CPU and memory utilization of an Apache web server. They design a modeling agent that builds a dynamic system model from the controlled server data. A controller design agent that uses optimal control theory to derive a feedback control algorithm customized to that server; and a run-time control agent that deploys the feedback control algorithm in an on-line real-time environment to automatically manage the web server. The designed autonomic feedback control system is able to handle the dynamic and interrelated dependencies between the tuning parameters and the performance metrics.

Andrew Lee, Mohammed Ibrahim, et al. [5] describe the principles behind the autonomic computing system explored how an implementation can be used to demonstrate autonomic behavior. It devised a simulation experiment to determine the mathematical relationship used to calculate attribute values and the criteria used when actions are triggered.

From the point of view of fundamental autonomic element, these work are very instructive. But it did not clarify the autonomic element design and difficult to build a practical autonomic system.

3. Mind Agent Model for Autonomic Element Design

3. 1 AE Functional Structure

AE(autonomic element) is the basic element of autonomic computing system. AE collects the environment information, makes decisions and adjusts itself dynamically.

From the point of logic view autonomic element usually comprises the components of analyzing, monitoring, planning and executing, also contains the knowledge, sensor and effector components. Usually it is divided into four parts according to the functions:

Monitor: it is responsible for collecting, aggregating, filtering, managing and reporting information. The system should have uniform resource model in order to monitor and collect information consistently.

Analyze: it analyzes the complex environment and builds the model, which learns from the environment and predicts the future.

Plan: it constructs the action sequences according to the policy and agreement.

Execute: it executes the actions and manages the process control.

Figure 1 illustrates the autonomic element function structure. Four parts work together to provide the self-management capability by policies.

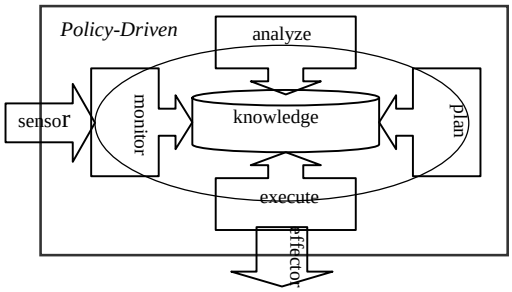

Fig. 1. Autonomic element function structure

The ring connecting the respective parts should be looked as a message bus, not the strict control flow. Four parts collaborate to complete the specified task via communication.

In order to adapt itself to the changing environment and solve tasks cooperatively, AE must use the knowledge to update the internal mind states that provide a further plan for how to take actions.

3.2. AE Lifecycle

Autonomic system is different from general software system and autonomic element varies from the traditional component. Autonomic element should consider self-management character from the start of design. The lifecycle of AE has the states depicted as figure 2.

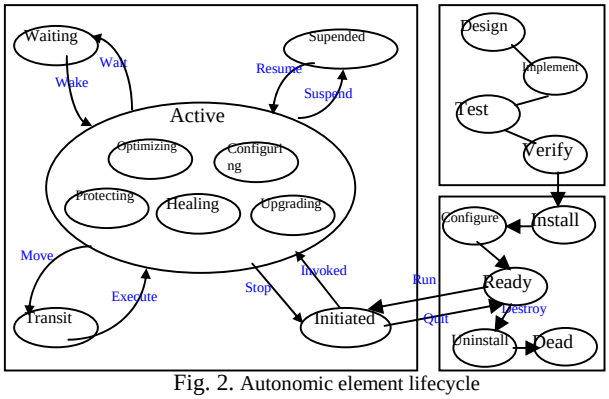

Fig. 2. Autonomic element lifecycle

When we deploy and configure autonomic element, the relevant information should be registered to the directory facilitator. Then autonomic element will enter into the ready state. Autonomic element will initialize itself once executing, then moves to the active state. If autonomic element pauses, it will transfer to suspended state. AE can come back to active state when resumed. If it does not meet the running conditions it will switch to waiting state. While migrating to other nodes AE will transfer to the transit state. The active state comprises some sub-states including optimizing, configuring, protecting, healing, upgrading states and so on. Once AE quits, it will change itself to ready state. If it is uninstalled or destroyed, AE will jump to dead state [8].

Test and verify are used to ensure the correct autonomic element behavior. However in a large complex environment especially in distributed network, it is very difficult to test and verify the correctness. One feasible approach is to embed the test method into the autonomic element.

3.3 Mind Agent Model for AE

The mind agent model is defined as a six-tuple <K, A, G, P, I, L>, where

K is the belief knowledge base
A is the set of behavior capabilities
G is the set of goals
P is the set of plan
L is the set of policies
I is behavior intention

Belief is the most fundamental and important element of mind agent. Other mind states representation and reasoning rely on the rational believes. The belief can be looked as knowledge base including the basic axioms, the facts and the relevant domain data.

Behavior capability describes the capabilities of mind agent and decides what actions to take, what effects will be produced after taking actions. And the capability is the basis of the mind agent plan, which decides whether the goal can be achieved.

Policies including action policy, goal policy, utility policy or mixture policy, describe the rules to guide the system behaviors [8].

Goal reflects the state or behavior changes after executing a specified task, which is usually produced when it specifies the tasks and policies.

Plan is the action plan and determines the approaches to achieve the goals. Plan connects the belief, the capability and goal together, which illustrates what actions to take for completing the specified goal based on the belief.

Intention is not equal to goal. Some scholars look intentions as the choice and commitment of plan, however, in our model the result of the plan, namely the action sequence, is the behavior intention of the autonomic element.

The architecture of mind agent is shown in figure 3.

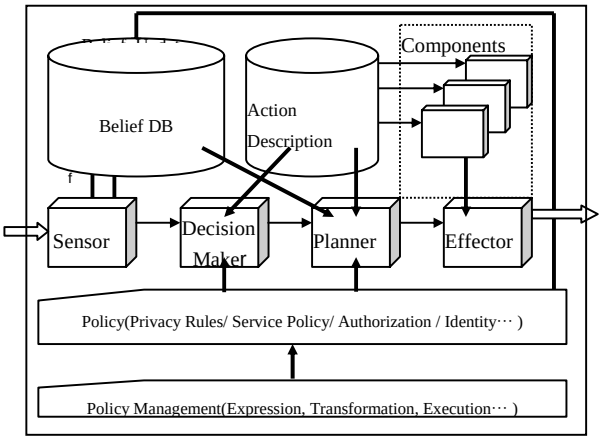

Fig. 3. Mind agent model

The belief base is the bases of knowledge representation and reasoning.

Action descriptions are used to express the capabilities and services of a mind agent. Each action description is connected to a behavior entity, which is encapsulated as a component. These behavior entities act as functional components with standard interface and can be loaded dynamically.

Sensor can be regarded as the sense organs of mind agent. It can observe the environment and receive messages from other mind agents or users. When sensing new occurrences, the sensor will send the new observation to decision maker, also it can update the belief of mind agent.

Decision maker can be regarded as a reasoning machine, which will decide whether a new goal will be achieved or not. The decision will be made based on mind agent belief, observation, and policy. There are three levels of decision. The first level is directly made based on observation, which is called reflection decision. The second level of decision is based on mind agent belief and policy, which is called intelligent decision. The third level of decision is automatically made based on mental state of mind agent, such as desires, long-term goals and complex policies.

After a new goal is added into the set, it denotes that the mind agent has an intention to reach the goal. Then planner will decide how to achieve it. The planning algorithm is based on mind agent belief, action description and policy. A static plan can be generated quickly by specifying plan algorithms. But dynamic plan is more adaptive and flexible according to complex policies. The actions of a plan will be added into the action queue if the plan is successful. The effector will load behavior entities to execute and impose on the outer environment.

A mind agent program will repeatedly execute the following steps:

(1) Observe the environment and update agent beliefs;

(2) Generate goals according to beliefs and observation of agent, then add it into G;

(3) Select one of the most important goal $\delta \in G$ to be achieved;

(4) Find a plan ($\alpha 1,\ldots, \alpha n$) to achieve goal δ according to the policy;

(5) Execute the plan and feedback to agent belief.

Based on the mind agent model the autonomic element structure is defined as Figure 4.

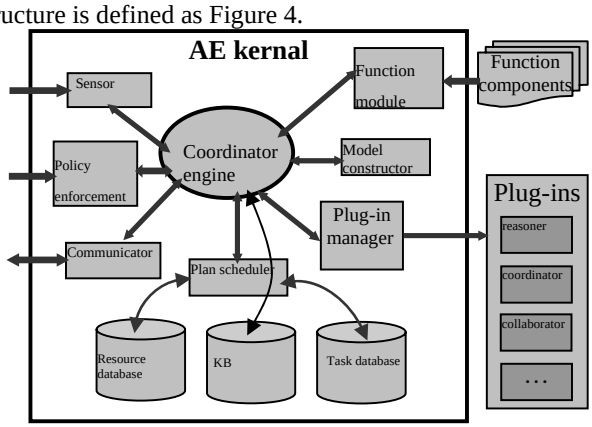

Fig. 4. Autonomic element structure

Sensor senses the environment and collects the needed information.

Function module interface acts on the environment and performs actions.

Communicator is used to communicate with other elements, improving the interoperability.

Model constructor analyzes the complex environment and builds the system collaboration model.

Policy enforcement point executes the policy decisions, monitors and feeds back the running process of decision. Autonomic computing system should have the module to store, manage policies and make decisions based on the dynamic policy.

Plan scheduler is in charge of scheduling and performing the actions to complete the task based on the constructed models and the action sequences.

Coordinator engine coordinates the interoperability, communication and collaboration of different modules, maintaining the autonomic element operating. When some module fails, the coordinator engine can restore it such as re-loading the new function module to replace the failing one.

Resource database stores the available resource list of autonomic element.

Task database provides the action logic description.

Knowledge base provides essential information (belief etc.) for the management.

Plug-in manager manages the provided plug-ins such as reasoner, coordinator, and collaborator so on.

3.4. Self-optimization

Autonomic system is made up of a group of the inter-dependent and interconnecting autonomic elements. An autonomic system s can be described as $s =< A, f >$, where $A = \{a_1, a_2, ..., a_n\}$ is the set of autonomic elements, $f : a \in A \rightarrow 2^A$ is the dependency function mapping an autonomic element to the autonomic element set. When we say a is dependent on b, it means that a needs the service b provides. $f(a)$ comprises the providers a needs, namely, $f(a)$ is the set a depends on, a has the connection with the autonomic element in $f(a)$.

Considering the autonomic element a_1 and a_2, they rely on the b_1 and b_2 respectively. Now assuming that the final result of a_1 vs b_1 is sat_{11}, and that of a_2 vs b_2 is sat_{22}. If we change the connection a_1 vs b_2, a_2 vs b_1, their final result is sat_{12} and sat_{21} respectively. If $sat_{12} + sat_{21} > sat_{11} + sat_{22}$, we say the result is superior to the last one[8]. There is an assumption that b_1 and b_2 provide the same service for a_1 and a_2.

Because every element has an opinion about the provided service, such as service efficiency, service cost, service quality and so on. We can look at the final result as the comprehensive evaluation of the provided services. The higher final result is, the better the self-optimization is. Then we introduce $\mathrm{SAT}(a,b)$ to represent the final result a vs b.

Definition: Assuming that at the moment of t, the autonomic system s has n autonomic elements, $s =< A, f_1 >$, $A = \{a_1, a_2, ..., a_n\}$, f_1 is the dependency function, P is the total number of the policies specified in s, $SAT_1 = \sum_{i=1}^{n} \sum_{\forall a_j \in f_1(a_i)} sat(a_i, a_j)$. If at the moment of $t+l$, according to the new policy or business goal the autonomic system changes the autonomic elements connections. And the function is changed to f_2, $SAT_2 = \sum_{i=1}^{n} \sum_{\forall a_j \in f_2(a_i)} sat(a_i, a_j)$. If $SAT_2 > SAT_1$, the connection change can be looked as the self-optimizing process.

Therefore the self-optimization of autonomic system can be looked as the reasonable service allocation, that is, it is the process of selecting better services to decide the better autonomic element connections. If an autonomic element a needs some kind of service, it will query the autonomic directory facilitator (ADF) and request the service. Then ADF will choose a suitable autonomic element b for a, a will use the service b provides. If a is unsatisfactory with the service it will request the ADF for another provider.

4. Application

The autonomic element based on mind agent model is tested in the multi-agent platform MAGE[9] developed by key laboratory of intelligent information processing, institute of computing, Chinese academy of sciences. MAGE is a multi-agent development and running environment, which is based on the intelligent and multi-agent technologies, utilizes the agent-oriented software engineering to analyze, design and deploy the multi-agent system. As a distributed development platform, the MAGE environment comprises three tools: Modeling Tool for Multi-Agent System supports the system analysis and design including AUML modeling tool; Visual Agent Studio(VAStudio) is a multi-agent development platform; MAGE deployer provides the deployment and running environment for the multi-agent system[10].

AGrIP(autonomic grid intelligent platform) is developed on the basis of MAGE. Our goal is to make full use of agent and autonomic computing technologies to build an autonomic grid test environment for the applications. Because autonomic computing is an evolutionary process AGrIP are evolving from the agent grid intelligent platform to autonomic grid platform. Compared with the previous design [11] AGrIP not only makes full use of the agent technologies but also incorporates the autonomic computing advantages from the kernel design to high level implementation.

In the past decade attempts to apply intelligent agents in realizing the grid vision have been made by academic researchers. The most interesting work in the literature might be the Agent Grid concept proposed under the DARPA ISO's Control of Agent-Based Systems (CoABS) program. The agent grid is a specific mechanism for sharing distributed services and resources. We can view the grid as a number of interacting agent-based autonomic nodes. And we built different grain-level policy plug-ins from the autonomic element, node, middleware to the application level. Figure 5 shows the AGrIP architecture.

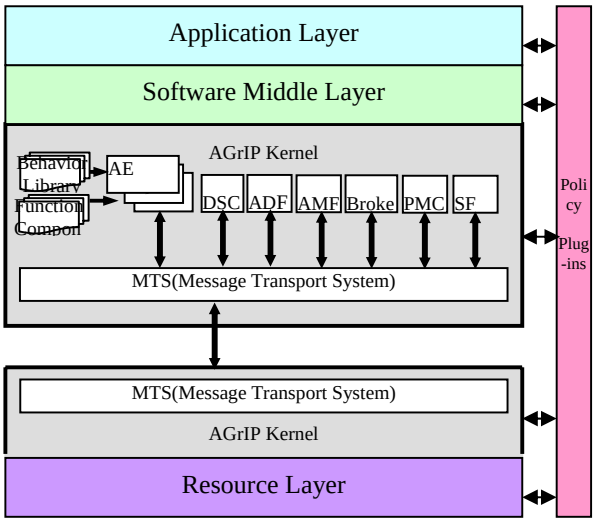

Fig. 5. AGrIP architecture

AGrIP makes it possible to configure autonomic element dynamically through deployment solution center(DSC). Using address mapping facilitator(AMF) autonomic element can communicate via name directly so as to eliminate the address complexity. Using autonomic directory facilitator(ADF) we can provide self-configuring and self-optimizing based on services. Broker has the abilities of service composition and coordination. While policy management center(PMC) is responsible for the policy management. MTS provides the transparent communication. Furthermore autonomic inherits the agent

characteristic of coordination and reasoning. Also security issues are very important in autonomic systems. So security facilitator(SF) answers for security detection. AGrIP is a collaborative environment and provides a resource-sharing platform. It can manage the sharing resource of the grid nodes, including hardware, data, information, knowledge and services etc, where the hardware resource includes the grid node CPU, memory, storage and their utilization. And services are the agent services developed by VAStudio[12] and web services, which provide the data mining, machine learning abilities etc. After the node resources are updated AGrIP will update itself timely and coordinate the resources.

Figure 6 shows the AGrIP management GUI. It describes the grid node GUI that includes two dynamic physical nodes *qiu* and *huj*. It can show all the relevant nodes if they registered to AGrIP.

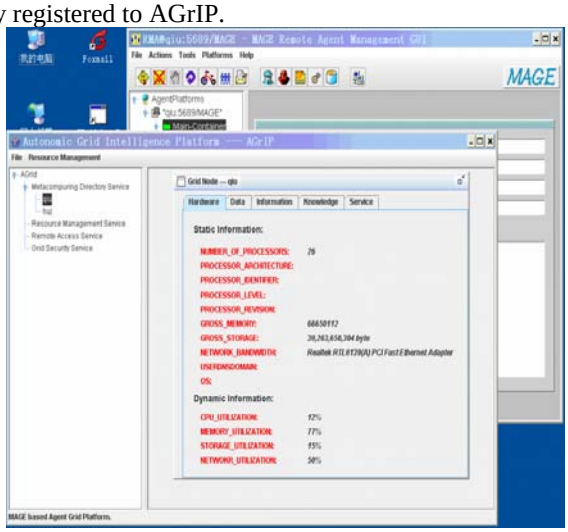

Fig. 6. AGrIP management GUI

Every registered node manages the hardware and software resources, which can collect data, extract information, manage knowledge and provide services so on. Once the task is distributed to different autonomic nodes, every grid node will monitor the network resource and local hardware utility including CPU, memory, disk and network resource to make decisions based on the concrete policy. When distributing the sub-task to the nodes according to the policy, every node will calculate the resource utility and the provided services, then allocate the sub-tasks to the optimal nodes. If the running task in one grid node such as *qiu* is interfered by the computer or the user, it will monitor and determine which registered node is most suitable according to the policy, then it will migrate to the node such as *huj* and continue to complete the work otherwise it will have to wait. Ultimately the related nodes will collaborate to complete the complex work. The grid node has proved that it has the advantages of autonomic element and agent simultaneously.

AGrIP has been applied successfully in a complex, dynamic, multi-agent domain, such as GEIS(Grid-based Emergency Interaction System), distributed data mining and so on[12].

5. Conclusion and Future Work

The autonomic element based on the mind agent not only has the traditional agent advantages but also incorporate the policy to make decisions. It is a promising approach to meet some challenges in autonomic computing initiative. Further we are focusing on the complex collaboration and consistency checking among the different nodes in AGrIP.

Acknowledgments

This work is supported by the National High-Tech Programme of China (Grant No. 2003AA115220), the National Basic Research and Development Plan of China (Grant No. 2003CB317000) and the Youth Research Programme of China University of Mining and Technology (Grant No. 0D4489).

References

[1] Jeffery O. Kephart, David M. Chess. The Vision of Autonomic Computing Outlook. IEEE Computer Society, January 2003, pp.41-47.

[2] A.G.Ganek, T.A.Corbi. The dawning of the autonomic computing era. IBM SYTEMS JOURNAL, VOL42, NO 1,2003, pp.5-18.

[3] J.P Bigus, D.A.Schlosnagle, J.R.Pilgrim, W.N.Mills III,Y.Diao. ABLE: A toolkit for building multiagent autonomic systems, IBM SYTEMS JOURNAL, VOL 41,NO 3, 2002, pp.350-371.

[4] Y.Diao, J.L.Hellerstein, S.Parekh, J.P.Bigus. Managing Web server performance with AutoTune agents. IBM SYTEMS JOURNAL, VOL 42,NO 1, 2003, pp.136-149.

[5] Andrew Lee, Mohammed Ibrahim. Emotional Attributes in Autonomic Computing Systems. Proceeding of the 14th International Workshop on Database and Expert System Applications(DEX'03).

[6] Mingkai Dong. Research on Dynamic Description Logic for Intelligent Agent, PhD Dissertation. Chinese Academy of Sciences, Beijing, 2003.

[7] Mingkai Dong, Haijun Zhang, Zhongzhi Shi. The Agent Model Based on Dynamic Description. Journal of Computer Research and Development.2004, 41(5), pp.780-786.

[8] Haijun Zhang. Research On Agent-Based Autonomic Computing. PhD Dissertation. Chinese Academy of Sciences, Beijing, 2005.

[9] Zhongzhi Shi, Haijun Zhang, Yong Cheng, Yuncheng Jiang, Qiujian Sheng, Zhikung Zhao. MAGE: An Agent-Oriented Programming Environment. IEEE ICCI 2004, pp. 250-257.

[10] Maoguang Wang, Zhongzhi Shi, Jiewen Luo,et al.Dynamic Interaction Protocol Load in Mult-agent System Collaboration. PRIMA2005,pp.101-115.
[11] Maoguang Wang, Jiewen Luo, Fen Lin, Zhongzhi Shi. Internet intelligent Platform--AGrIP. Artificial Intelligence Applications and Innovations II(AIAI2005), pp.363-370.
[12]Intelligent Science Web site: http://www.intsci.ac.cn[OL],2006.