Proceedings of the 6th World Congress on Intelligent Control
and Automation, June 21 - 23, 2006, Dalian, China

A Self-Organized Model of Agent-Enabling Autonomic Computing for Grid Environment

Hang Guo[1], Ji Gao[2], Peiyou Zhu, Fan Zhang

Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027, Zhejiang, China

E-MAIL: gentleguoh@yeah.net[1], gaoji@mail.hz.zj.cn[2]

Abstract - **Agent-enabling autonomic computing is a promising solution to the system management troubles caused by increasing complexity of large scale, distributed systems. This paper presents a self-organized model of agent-enabling autonomic computing for Grid environment. This new model adopts intelligent agents as autonomic elements and enables these agent-based elements to dynamically organize system management works without centralized control or directions. In the element level, each agent possesses some capabilities and interests based on its managed resources, and governs internal affairs to achieve elementary autonomy. In the system level, they contribute their capabilities on the functions of system management, and cooperate together to implement autonomic computing for system. Cooperative works are organized by dynamically associated relationships among autonomic elements (agents), including acquaintance, collaboration and notification. The self-organized model is more suitable for such environments as Grid, which is characterized by distributed, open and dynamic properties.**

Index Terms – Autonomic Computing, Multi-Agent, Grid.

I. INTRODUCTION

In recent years Grid, which facilitates the sharing and integration of large scale, heterogeneous resources, has been widely recognized as the future framework of distributed computing [1]. However, the increasing complexity of Grid services and systems demands correspondingly larger human effort for system configuration and performance management. The management issues, which are mainly done in a manual style today, become time-consuming, error-prone and even unmanageable for human administrators. Autonomic computing [2, 3], presented and advocated by IBM, suggests a desirable solution to this problem. The vision of autonomic computing is to design and build computing systems that possess inherent self-managing capabilities. Each autonomic system is a collection of autonomic elements, which are elementary constituents that possess services and resources. By self-configuration, self-optimization, self-healing, and self-protection, autonomic computing can improve the level of automation and self-management capability to a far greater extent than it is today in Grid computing systems.

To achieve the vision of autonomic computing, many researchers advocate multi-agent system (MAS) [7] as the most suitable approach of constructing autonomic systems [4, 5, 6]. They support the idea that autonomic elements can be seen as agents and autonomic systems as multi-agent systems. In current agent-based autonomic computing architectures, the self-management process is mainly formed and organized by static control and directions. Selected agents are in responsibility for certain management works, and the cooperation procedure of multi-agents for accomplishing self-management is established by human design, policy or predefined strategy. However, the static organization mode of self-management has shortcomings in open, dynamic and heterogeneous Grid environment: 1) Centralized control may be invalid or difficult to organize heterogeneous agents, which have different properties and interests, and also conflicts to the decentralization characteristic of Grid. 2) It is not robust in such open system, because some agents who are in responsibility for management works may dynamically exit or change their capabilities, thus the static organization may be failed. 3) Unpredictable events and requirements cannot be well coped with by static strategies in such dynamic and complicated environment. Hence, it needs more flexible organization mechanism to implement self-management capability of autonomic computing in Grid environment.

In this paper, we propose a self-organized model of agent-enabling autonomic computing for Grid environment. In this model, autonomic elements (agents), which manage several management-related Grid services, dynamically organize management works without centralized control and directions. Autonomic elements establish *acquaintance* relationships to acquire each other's *capability* and *interest*. Based on such acquaintance information, they are able to form collaborations by their interactions and contribute each member's capability to accomplish necessary sub-tasks when management is needed. Every participant acts according to its capability and knowledge, and send results, further request and information to others for cooperative work. They as a whole present a self-organized mode to achieve the self-management capability, required by autonomic systems. This self-organized model avoids the weaknesses of current statically controlled models, and is more suitable for Grid environment. This model is supported by a prototype multi-agent platform that has been developed in many years and extended to a Grid environment recently [8, 9].

In following sections, the architecture of the model is introduced in section 2, the agent-managed autonomic elements and the relationships among them are represented in section 3 and section 4 respectively, and the self-organized mechanism is represented in section 5. Finally we draw conclusions in section 6.

1-4244-0332-4/06/$20.00 ©2006 IEEE

II. Architecture of Self-Organized Agent-Enabling Autonomic Grid System

In the model, agents are autonomic managers of autonomic elements, governing several Grid services and lower-level agents. Agent governs internal affairs including service management (registration, invoking, monitoring, and exception handling) and coordination of lower-level agents, as well as external affairs for interaction and collaboration. It is distinguished two types of Grid services: *management services* which are Grid services used for system management works, and *application services* which are those services applied for human demand. Management services are the elementary function units constituting system management works. The implementation of a self-management task can be seen as a sequence of management service executions controlled by autonomic elements (agents). The structure of the agent-managed autonomic element is shown in figure 1.

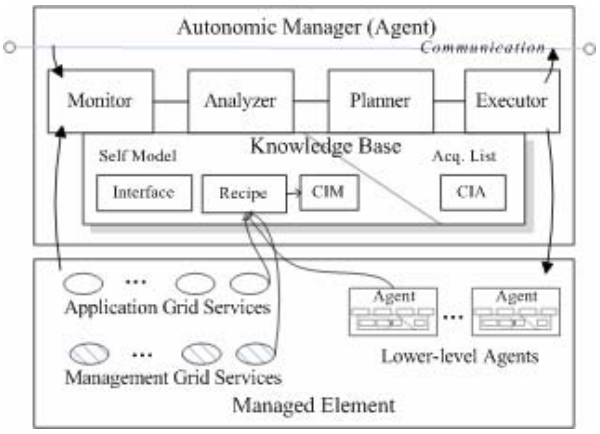

Fig. 1 Structure of agent-managed autonomic element.

Composed of agent-managed autonomic elements, autonomic system can be seen as a multi-agent system that enables autonomic computing. Each agent publishes and advises its *capability* and *interest* to other agents, including medi-agents which mediate matchings between service request and provision, and records such information of others. Capability is what an agent is able to perform, and interest is which information an agent considers important for further actions. When a system management work is needed, agents transmits useful information and task requests to others according to the capability and interest information, constituting a multi-agent collaboration to carry out the management work. Each participant in the collaboration uses its managed elements (services, lower-level agents) to execute one or more sub-tasks, and they together perform a self-management work. Hence the process of a self-management work is self-organized by agents, and is dynamic and flexible against changeable contexts. The architecture of the self-organized agent-enabling autonomic Grid system (SAAGS) is shown in figure 2. SAAGS can be defined as a 3-tuple:

$$SAAGS ::= (AE, ER, SOM) \quad (1)$$

AE – set of autonomic elements.

ER – set of relationships between AEs.

SOM – self-organized mechanism of agent-enabling system management.

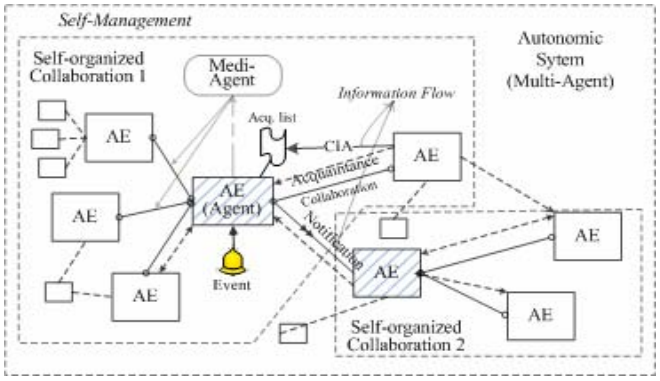

Fig. 2 Architecture of the self-organized agent-enabling autonomic Grid system.

III. Agent-Managed Autonomic Element

From the view of autonomic computing, autonomic element (AE) is composed of one autonomic manager and several managed elements. In our model, the autonomic manager of each autonomic element is an agent, and managed elements are application Grid services, management Grid services, and/or lower-level agents (Fig. 1). The manager agent integrates services and lower-level agents to perform high-level functions, and undertakes management works for managed elements in their whole life cycles. External interactions, such as information exchange and cooperative work, are also carried out through agents' communication interfaces.

Definition 1. Autonomic element is defined as a 3-tuple:

$$AE ::= (AM, ME, IR) \quad (2)$$

AM – autonomic manager, which is an agent.

ME – managed elements, which can be application and management Grid services, and/or lower-level agents.

IR – internal relationships between the AM and MEs.

Relationships in IR are mainly described in the *self-model* (SM), which indicates how the manager agent integrates, publishes, invokes, monitors, and controls managed elements. Self-model mainly consists of three parts: *Recipe*, *Interface*, and *CIM (Capability and Interest Model)*.

In an autonomic element, managed elements (services, agents) should be integrated to higher-level functions and services that are closer to applications and cooperation. Such services are called *composite services*. The integration is transparent outside the autonomic element so that the composite services are published and performed as normal Grid services. *Recipe* is the guideline of the integration to implement composite services. It can be defined in the form of BNF as follows:

Definition 2.

Recipe := { <PlanSteps> | (loop<PlanSteps>) }$^+$

<PlanSteps>:={ (←{return | <OperCall>} <Cond>)

| (← <OperationSet> [<Cond >])

| (or{←<OperationSet> [<Cond >] }$^{+}$) }$^{+}$

<OperationSet> := ({sequence | concurrency}{← <OperCall> [<Cond >]}$^{+}$)

<OperCall> := ([<Agent>:]<Service>.<Operation> {<Parameter>}*)

<Cond > := <Condition-Expression>

The form of the Condition-Expression in recipe should conform to specific syntax frameworks. The services and operations called in <OperCall> may be controlled by other AEs (agents), requiring a collaboration to implement the composite service. Recipe, which can be established manually or autonomically, provides a flexible approach to extend managed elements to various application-oriented composite services, and thus make agent able to undertake more complicated tasks.

While a recipe directs how to accomplish a complicated task (composite service), *interface* describes how to use and control managed elements, such as service invoking, monitoring and handling. Without such description, agents cannot dynamically invoke and manage low-level computing resources and components. The form of interfaces may be varied according to different platforms. In a OGSA-compliant Grid platform [10, 11], the main form of interface should be WSDL with semantic extensions.

In a complex system, many management works need collaboration to be accomplished. To this end, autonomic elements need to publish or advertise what they can do and which information it takes interest in. *Capability* and *Interest* are abstracted from managed elements and recipes by manager agent to describe such information. The representation of capability and interest is based on ontology and first order predicate logic, enabling semantic cooperation and logic reasoning. The formalized definitions of them are defined in following.

Definition 3. *Capability* presents what tasks an autonomic element is able to undertake for external affairs. It can be defined as a 3-tuple:

$$Capability ::= (Category, Constraints, Params) \quad (3)$$

where *Category* denotes the specifically defined class which the capability is classified to, *Constraints* denotes the specific conditions for application of the capability , and *Params* describes the parameters which should be inputed when invoking the capability. Their BNF-form definitions are given:

- <*Category*> := (ClassificationNo {<CatAspectName> Category-name}$^{+}$)
  <CatAspectName> := { "GeopoliticalRegion" |"General Category" | "ProvisionCategory"}
- <*Constraints*> := { <Concept-Pattern> }$^{+}$
  <Concept-Pattern> := ([<Ontology>:]<Concept>{<Slot-Pattern>}$^{+}$)
  <Slot-Pattern> := (Slot-name, {value | <Var-Constraint > | <Concept-Pattern>})
  <Var-Constraint> := (Var-name, <Predicate-Formula>)
- <*Params*> := ("In-Params:"{<DataType>}$^{+}$, ["Out-Param:" <DataType>])
  <DataType>:={ base-type | [<Ontology>:]<Concept> }

Definition 4. *Interest* indicates the information that an autonomic element is interested in. It is what the manager agent considers useful for the decision of start-up and processing of tasks. The representation of interest is based on ontology with first order predicate logic embedded, enabling the matching between interest and information by reasoning.

$$Interest :: = \{ [\&] \text{<Concept-Pattern>} \}^{+} \quad (3)$$

where <Concept-Pattern> has been defined in definition 3.

Manager agent collects capabilities and interests to establish *Capability and Interest Advertisement*, *CIA* ::= ({*Capability* }$^{+}$,{ *Interest* }$^{+}$), which will be advertised to other agents and published to medi-agents. It is correspondingly needed the mappings from capabilities and interests to internal implementation for them, such as recipes, services, functions, policies. CIA and the mappings constitute the *Capability and Interest Model*.

Definition 5. *Capability and Interest Model* is defined as a tuple:

$$CIM ::= (CIA, \text{Mappings}) \quad (4)$$

CIA –Capability and Interest Advertisement.

Mappings – mappings from those elements in CIA to the corresponding methods of internal implementation.

IV. Relationships between AEs: Acquaintance, Collaboration and Notification

Self-organized management work in an autonomic system demands information sharing and cooperation between AEs. Relationships between AEs are formed correspondingly. Three main relationships among them are *acquaintance, collaboration* and *notification* (Figure 3).

When an autonomic element receives the CIAs published by others, it records these CIAs and their publisher AEs as acquaintance information. In other words, given two autonomic elements A and B, A is B's *acquaintance*, if B knows A's capabilities and interests. B can request the services that A has capability to do or notify A with its interested information when needed. It is the foundation of self-organization management to establish and maintain acquaintance relationships, because cooperative works are based on them. The list that stores the acquaintance information in the knowledge base is called *acquaintance list*. Acquaintance lists can be exchanged between AEs, and hence CIAs be further transmitted. The publishing and transmission mode of CIAs and the establishment and update mechanism of acquaintance lists are decided by the organization form and strategies of specific autonomic system.

C*ollaboration* relationships between AEs are established to accomplish cooperative works. In collaboration, each element undertakes partial sub-tasks and works jointly with other members to fulfil the collective goal. The life cycle of collaboration includes invitation, negotiation (not necessary), affirmance, request, monitoring/exception report, exception handling, and termination (Figure 3). There may be an initiator or organizer in a collaboration to invite participants, do coordination and handle exceptions, for instance, the AE who has the collective recipe. Multi-agent collaboration has been

studied for many years [7, 12]. The practical challenges of collaborations for agent-enabling automatic computing may likewise spur basic research advances within the MAS community.

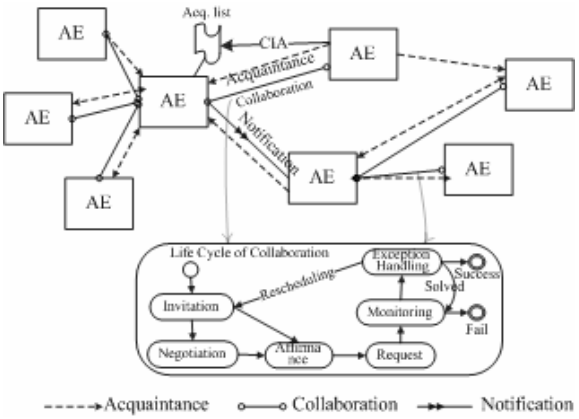

Fig. 3 Relationships between AEs.

When an element has some information that matches some other's interests, it can send a message containing the information to the interested element. In this condition, they have a *notification* relationship. Notification relationship is actually an information flow between AEs. Notification message helps the interested element doing analysis and processing or may trigger a management task. Different from collaboration, notification does not explicitly decide what will act, and the sender does not maintain a control. A key problem in notification is to decide whether a piece of information (or an *event*) "matches" an interest. The event is formalized as what is called *concept instance*:

<Concept-Instance> := ([<Ontology>:] <Concept> {<Slot-Value>}$^{+}$) (5)

<Slot-Value>:=(Slot-name,value | <Concept-Instance>) (6)

Definition 6. To decide whether an event *matches* an interest, it is verified by following rules:

1) If the concept instance of the event, <Concept-Instance> *matches* any <Concept-Pattern> in the interest (formula 3), the event *matches* the interest.
2) <Concept-Instance> matches <Concept-Pattern>, if they have the same *ontology* and *concept*, and every <Slot-Value> of the former *matches* the corresponding <Slot-Pattern> of the latter.
3) Given a <Slot-Value> = (Slot-name, $value_1$) and <Slot-Pattern> = (Slot-name, $value_2$). If $value_1$ = $value_2$, then <Slot-Value> *matches* <Slot-Pattern>.
4) Given a <Slot-Value> = (Slot-name, $value_1$) and <Slot-Pattern> = (Slot-name, <Predicate- Formula>). If $value_1$ satisfies the <Predicate- Formula>, which is a first order predicate formula, then <Slot-Value> *matches* <Slot- Pattern>.
5) Given a <Slot-Value> = (Slot-name, <Concept-Instance>) and <Slot-Pattern> = (Slot-name, <Concept-Pattern>). If <Concept-Instance> *matches* <Concept-Pattern>, then <Slot-Value> *matches* <Slot- Pattern>.

V. Self-Organized Mechanism of Agent-Enabling System Management

The vision of self-organized system management is to dynamically organize a management process, namely deciding whether a management task should be started, forming collaboration, and coordinating members' sub-tasks, by system elements themselves without centralized control and directions. Each autonomic element may not have complete knowledge or capability to realize what management processing is needed or how to carry out such complex procedure. It just contributes what it considers necessary and has capability to do, and sends further requests and information to other elements. However in the macroscopical view, associated by the acquaintance, collaboration and notification relationships between AEs, a dynamic association is autonomically formed and coordinated to carry out the complete management work. Self-organized mechanism enables autonomic elements to associate with proper relationships to achieve such vision. Following steps are complied with by each autonomic element, implementing agent-enabling autonomic management in a self-organized manner.

Step 1: Based on current information, manager agent decides what management work is needed. If none, go to step 9.

Step 2: If the manager agent knows how to do the needed management work (has a corresponding recipe), then go to step 6.

Step 3: Search in its acquaintance list which acquaintances have capability to do the management task. If none, go to 9.

Step 4: Select one in these candidate acquaintances, and send a request of the task to it.

Step 5. If the request has been executed successfully, stop. If there are no more candidate acquaintances, go to step 9; else go to step 4.

Step 6. If all sub-services in the recipe can be executed locally by the element, execute the management work under the direction of the recipe, and go to step 9.

Step 7. For each sub-service that need external provision, search acquaintances that have the corresponding capabilities (or request medi-agents if necessary) and invite them to participate in the collaboration. If any sub-service has no participant to undertake, go to step 9.

Step 8. Start the collaboration to carry out the management work. Monitor the execution of every participant, and cope with collaboration exceptions.

Step 9. If some current information matches some interests of acquaintances in the acquaintance list, notify those interested elements with such information.

VI. Conclusions

The increasing complexity of Grid services and systems makes the system management work pose great challenges. This paper proposes a self-organized agent-based model of autonomic computing to solve this problem. Intelligent agents are adopted as autonomic elements in this model. These agent-

based elements dynamically establish and maintain the acquaintance, collaboration and notification relationships between each other to accomplish system management works in a self-organized mode. The framework and principles of this model are described from the aspects of elementary constituent, relationships, and self-organized mechanism.

In future works, how to maintain relationships for effective self-organization, such as the transmission and synchronization mechanism of acquaintance lists, will be further studied. And performance experiments are also needed to improve the self-organized mechanism.

ACKNOWLEDGMENT

Our work is supported by the Major State Basic Research Development Program of China under Grant No. 2003CB317000.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", Int'l J. High-Performance Computing Applications, vol. 15, no. 3, 2001, pp. 200-222.

[2] Jeffrey O. Kephart and David M. Chess. "The Vision of Autonomic Computing", IEEE Computer, January 2003.

[3] A. G. Ganek，T. A. Corbi, "The Dawning of the Autonomic Computing Era", IBM Systems Journal, .42 n.1, p.5-18, January 2003.

[4] Tesauro, G., Chess, D.M., Walsh, W.E., "A Multi-Agent Systems Approach to Autonomic Computing", In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004, Page(s):464-471.

[5] Y. Diao, J.L. Hellerstein, S. Parekh, and J.P. Bigus, "Managing Web Server Performance with AutoTune Agents", IBM Systems Journal, Vol. 42, No.1, pp. 136-149, 2003.

[6] T. De Wolf, and T. Holvoet, "Towards Autonomic Computing: agent-based modelling, dynamical systems analysis, and decentralised control", In: Proceedings of the First International Workshop on Autonomic Computing Principles and Architectures 2003, pp.10.

[7] N.R.Jennings. "On Agent-Based Software Engineering". Int Journal of Artificial Intelligence,vol.177, no.2, pp: 277-296, 2000.

[8] GAO Ji, WANG Jin. "ABFSC: An Agents-Based Framework For Software Composition", In: Chinese Journal of Computers, Vol22, No.10, Oct. 1999, pp. 1050-1058.

[9] GAO Ji, YUAN Cheng-Xiang, WANG Jin. "SASA5: A Method System for Supporting Agent Social Activities", In: Chinese Journal of Computers, Vol.28, No.5, May. 2005.

[10] I. Foster, C. Kesselman, J. Nick, S. Tuecke "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[11] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "Grid services For Distributed System Integration", In: IEEE Computer Volume 35, Issue 6, June 2002 Page(s):37-46.

[12] J. E. Doran, S. Franklin , N. R. Jennings and T. J. Norman. "On Cooperation in multi-agent systems". In: The Knowledge Engineering Review, 12(3), pp: 309-314, 1997.