Towards Autonomic Computing Using Goal-Based Translation Strategy in Heterogeneous Distributed Environments

POOIA LALBAKHSH MEHDI N.FESHARAKI
Department Of Computer Engineering
Islamic Azad University, Science & Research Campus
Ashrafi Esfahani Highway, Hesarak, Tehran
IRAN

Abstract: This paper presents a new architecture to implement autonomic features in complex heterogeneous environments. This three-layer architecture model eliminates the need for a common shared language and heavy global standards by using semantic negotiations between different layers. For this reason, a collection of high level cognitive agents make specific policy based decisions and an adapting interface layer sends them to the system elements. Using the received decisions and state information, each element tries to achieve the decision (goal) by its capabilities and local knowledge. To provide such local knowledge and semantic based functionality, we focused on dependability and its attributes as the key core of local autonomous and how they can affect the initial goals: self-configuring, self-healing, self-optimizing and self-protecting. Local elements will accomplish the goal by using the received state information and its dependability diagram. Such goal-based translation strategy will provide a better scalability and flexibility for today and future heterogeneous changing distributed environments and interwoven systems.

*Key –Words***:** Autonomic computing, Self management, Distributed architecture, Dependability.

1. Introduction

Increasing the complexity of computing environments and the need for appropriate support and maintenance particularly in critical applications such as e-commerce, military operations and real-time services, forced us to handle new problems of complexity, scalability and automatic management. Moreover, the need for experts with extensive knowledge to cover all aspects of such systems, the costly system downtimes and the necessity of continues service delivery [1] define great management problems particularly in distributed heterogeneous environments with thousands of different components. Since we can not stop the growing process of digital computing systems, we have to find new ways of efficient management of this complexity. IBM introduced the self-management strategy to handle these challenges which led to autonomic computing or AC for short [2]. This new concept contains many important features that help us to manage this growing process. Autonomic computing is based on four main concepts, namely:

1. Self - Configuring
2. Self - Healing
3. Self - Optimizing
4. Self -Protecting

These four Selfs, form the body of an autonomic system. A Self-Configured system can configure its components automatically to adapt to different conditions. Self-Healing property will automatically discover, diagnose and correct faults and therefore, protects the system against downtimes and service failures. Self-Optimizing guarantees the optimal functioning of the system by monitoring and adapting the resources, and finally self-protecting feature deals with the attacks against the system which guarantees a safe and secure service delivery. These four concepts are integrated expressions that consist of many underlying requirements to justify the desired conditions. We call these four concepts as initial goals that can lead to achieving the final goal or self-management.

The paper will continue as follows:
The next session will review a detailed background of autonomic computing and related works. In

section 2 we will take a look at heterogeneous distributed environments and interwoven systems. Then we focus on our strategy named goal-based translation and its characteristics. High level cognitive agents and their semantic negotiations will be introduced as high level concepts of the strategy. After clarifying the role of agents and the goal of translation, we will describe the dependability diagram of local elements as the underlying key to achieve the initial and then final goal. Then, the relations between four main concepts and the basic dependability attributes will be described. Fig. 1 is a global view of the proposed architecture. The high level core has many semantic based agents that can collect state information and drain the important semantics from the overall state, and then deliver the goal-based semantics to the components. These components are different elements with different languages with a local knowledge base. This local knowledge will cooperate with the high level agents to manage the component and lead it to achieve the desired goal. This architecture will guarantee the scalability and flexibility of the system. However such model could be well suited to mach with the current and future open standards

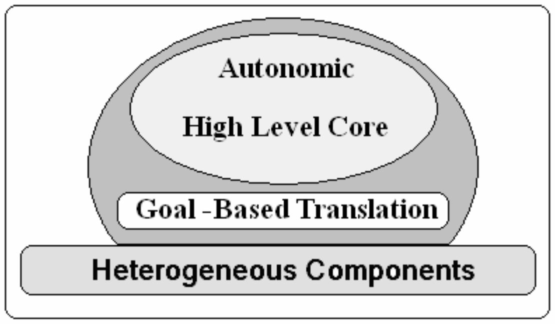

Fig. 1.Global view of the goal-based translation strategy

2. AC: Related Works and Basic Concepts

Although many people tried to introduce some of the features and characteristics of autonomic systems and their basic concepts, there is still a long way to implement a fully automated and self-managed system in reality. There is a large range of AC projects in academia and industry, but none of them could fulfill all of the aspects of autonomic environments. Here we take a brief look at some of the important AC projects:

- Unity project and autonomic computing toolkit [3] which tries to explore some of the behaviors and relationships that will allow complex configuring systems to self manage.
- KX project [4], an attempt to inject AC technology into legacy software systems without any need to understand or modify the code of the existing systems.
- Rainbow project [5,6], which investigates the use of software architectural models at runtime as the basis for the reflection and dynamic adaptation.
- ROC project [7], investigates novel techniques for building highly dependable Internet services. ROC emphasized recovery from failures rather than failure avoidance.
- DEAS project [8] that uses requirements goal models as a foundation for designing autonomic applications. Tasks such as self-configuration, self-optimization, self-healing and self protection are often accomplished by switching at runtime to a different system behavior.
- Astrolabe project [9], a new system to automate self-configuration and self-monitoring to control adaptation using a system-wide hierarchical database which evolves as the underlying information changes.

Many other AC projects are accomplished by different universities and some corporations [10] focusing on some specific challenges of these systems. Although many of them are nice solutions, we still need to have an efficient strategy to support the overall adaptation, flexibility and the four initial goals.

In the heart of each autonomic system there is a closed control loop with six main processes [2]: Sensing, monitor, analyze, plan, execute and effect. The sensors start the loop by collecting the appropriate information and the effectors finish It by effecting according to the execute process. Fig. 2 diagrammatically shows this closed control loop. Each of the intermediate processes or engines can communicate with the knowledge base. The monitor observes the data collected from sensors and then stores the distilled data in the knowledge base. The analysis engine compares the data against the desired sensor values also stored in the knowledge base. The planning engine devises strategies to correct the trends identified by the planning engine and the execution engine finally adjusts parameters of the managed element by means of effectors, and stores the affected values in knowledge base.

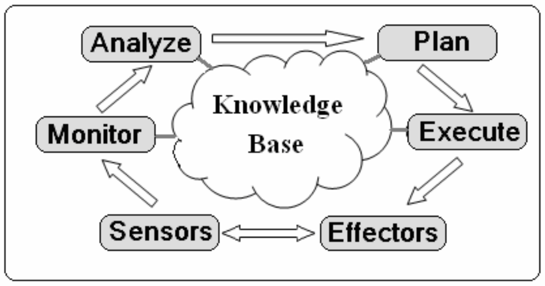

Fig. 2 Closed control loop of an autonomic system [2]

3. Challenges of Distributed Heterogeneous Environments

Converging computing systems with other applications increased the complexity and diversity of computing environments. On the other hand, this diversity introduces new challenges of maintenance and support tasks. Two important examples are using network based infrastructures in power-grids [11] and network centric operations [12]. Complexity and heterogeneity of these systems need a special management strategy based on that specific field. Using thousands of different components with different architecture models, negotiation languages and variety of vendors, needs an intelligent and instantaneous management that can not be manually accomplished. Fig. 3 shows a network centric military environment with different warfare components and sensors. These sensors and effectors have different functionalities and operational mechanisms. These ever changing heterogeneous environments can not be manually managed. Traditional management strategies would be time consuming and inefficient in such critical mission fields. Instead, an autonomic and efficient management system is required to accomplish the missions of such collaborative systems.

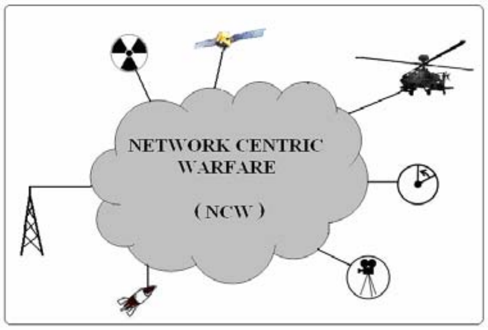

Fig. 3 A network centric military environment with different warfare components.

The growing use of distributed heterogeneous systems and infrastructures in different applications, forces us to answer many considerable management problems. Some of these problems are:

1. Maintenance and management of different components requires an extensive knowledge [13].
2. Because of heterogeneity, negotiations between components are difficult and sometimes a kind of manual translation is required [14].
3. Upgrade processes would be asymmetric, time consuming, costly.
4. Because of the complexity and heterogeneity, scalability is a great challenge.
5. Online repairs and optimization processes according to special policies are very complicated, time consuming or sometimes impossible [13].
6. The whole system is more than sum of its parts. This is the nature of a complex interwoven system [1].

Answering these problems efficiently is very important point in overall system performance. Autonomic computing is trying to achieve this improvement.

In some cases, like heterogeneous networks and heterogeneous multi-computer systems, an adapting layer is used to adapt system core, with other peripherals. For example in next generation networks, the soft-switch adapting layer is used to support flexibility [14]. This soft-switch adapting layer contains variety of functions to support different types of networked environment features.

These examples imply the importance of the adapting layer's rule in achieving an autonomic system. We will also use this idea in our architecture, together with some special strategies to gain the initial and final goals of the autonomic computing.

4. Goal-Based Translation strategy: a solution to heterogeneity and scalability

Translation into a common shared language, can improve portability and maintainability of complex heterogeneous systems [14]. But this strategy is not a comprehensive solution. To have a common shared language all the entities of the system should equipped with the language. It means that all the vendors should agree with a standard language. If we had such an agreement, we did not have any problem with heterogeneity, better to say: we did not have any heterogeneous system at all.

The presented examples and current applications show that, there is still a long way to have such an extensive agreement. On the other hand, preparing such global standards for today interwoven systems is a very complicated challenge [15]. To face these challenges, we proposed a goal-based translation strategy in such systems. In this method, the main goal is inferred from the higher levels and delivered to the different components of the system. The components then translate the final goal to their local domain to accomplish the requested job. In this architecture, not all the instruction should be translated, and the desired goal is only inferred to lower layers. All negotiations between layers are semantic based and therefore can guarantee the comprehensiveness and scalability of the overall system without imposing a common shared language.

4.1 A Layered model for heterogeneous systems

To describe the goal-based strategy we first introduce a layered model for a distributed heterogeneous system. Fig. 4 shows a three-level model with the following functionalities:

The highest level contains a collection of cognitive agents. These agents are aware of the overall system state and important policies that should be considered in the functional phase of the system life cycle. These agents can sense performance parameters of the system and then select the best goals related to the policies. The functionality of this multiagent layer is based on the four initial goals of autonomic computing.

The second layer is a simple layer only to communicate with the base layer. This layer receives the decided goal from the highest layer and sends it to the base layer. This layer works as an adapting interface.

The base layer is the local layer. This layer may contain different components with different languages. These components do not have any information about the upper layers or the overall system state and policies. The local layer should be able to accomplish the decided goals from knowledge layer, by means of local decisions.

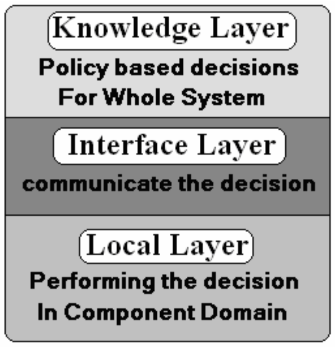
Knowledge Layer
Policy based decisions For Whole System
Interface Layer
communicate the decision
Local Layer
Performing the decision In Component Domain

Fig. 4 Layered model, of an autonomic heterogeneous system

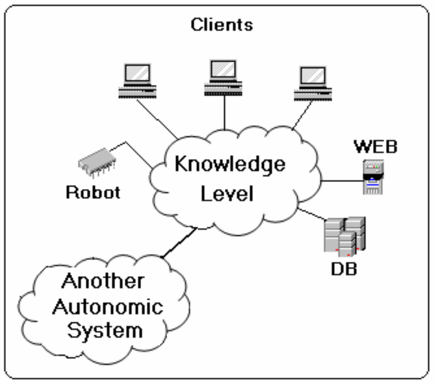

Fig. 5 An autonomic distributed system

Fig. 5 shows an autonomic distributed system. Many different components are connected to a network. The cognitive agents together with the special knowledge bases form the knowledge level of the system. These agents are monitoring the overall system performance and critical states of the system. The knowledge level is responsible to guarantee the four initial goals: Self-configuring, Self-healing, Self-optimizing and Self-Protecting. The cognitive high level agents can not directly conduct the peripherals. They just ask them to increase or decrease some of the performance parameters. Therefore, sensing, analyze and monitoring processes of the closed control loop is done by the cognitive agents in the knowledge level. After accomplishment of these processes, the needed goal will be determined. The selected goal will be send semantically to the component and then the component acts on it through its effectors. The context of each goal is defined for each component by its local functions and language. So the overall system knows same contexts but in different language. For example, in fig. 5 there are many different nodes on the network, each with different languages. For the whole system the meaning of self-healing is detecting and correcting the faults. But achieving self-healing in each node consist a different process. Therefore an identical self healing process can not be selected for both web server and robot.

4.2 Dependability Diagram: the Low Level Planning

After defining basic concepts and taxonomy of dependable and secure computing by Avizienis and his colleagues, dependability evaluation earned an important role in critical mission operations and related systems. Dependability and its attributes can define important states of a system. The main attributes of dependability is described in Fig. 6.

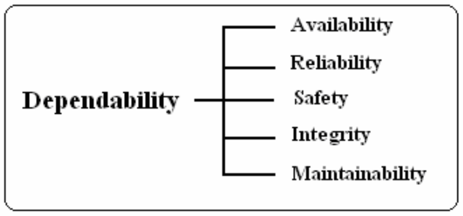

Fig. 6 Main Attributes of Dependability

Dependability can be considered as the base of each autonomic system. Most of the initial goals of each autonomic system rely on the dependability and its attributes. For example, a self-healed system needs high levels of availability and reliability, or a self-configured system would have a permissible maintainability level. Evaluating dependability and its attributes for any system is simple and follows a special rule. So we can use these known concepts to earn semi-known goals of the autonomic nature. The small definitions of the main attributes are [16]:

- Availability: readiness for correct service.
- Reliability: continuity of correct service.
- Safety: absence of catastrophic consequences on the user(s) and the environment.
- Integrity: absence of improper system alternations.
- Maintainability: ability to undergo modifications and repairs.

These attributes are the atomic measures of each autonomic system, and each component should achieve the requested goal by improvement of these parameters. System vulnerabilities and threats can decrease level of dependability attributes and consequently the system performance. These vulnerabilities should be recognized for each system component.

Having dependability attributes, affecting threats for each attribute and the means to attain the overall dependability of a system, we can define dependability diagram for each system, subsystem or component. Most of this information should be prepared by the vendors. Fig. 7 shows a dependability diagram for an interconnection network. This diagram contains the dependability attributes, vulnerabilities and treatment strategies such as: preventive and tolerating strategies.

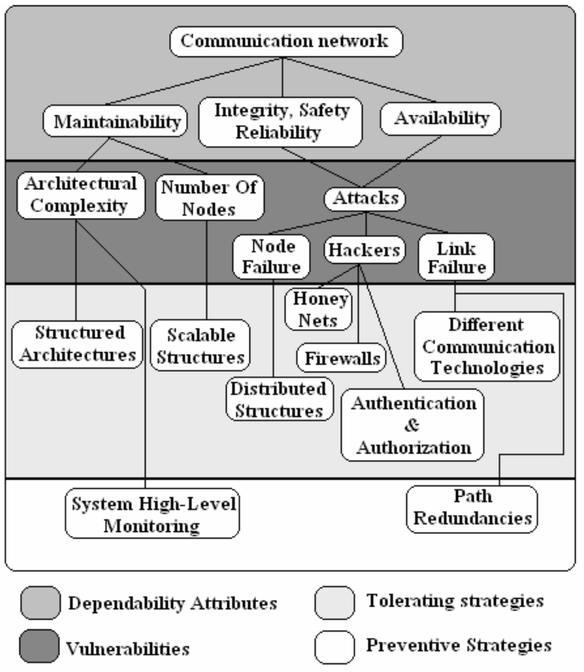

Fig. 7 Dependability Diagram for a Communication Network

The above diagram is summarized with the main points only to describe the main features of dependability diagram. The first level of the diagram contains the main attributes of dependability. You may add more attributes depending on the system you are working on. The second level is the threat level. This level should contain all of the possible threats against the system attributes. The third level consists of the tolerating strategies for each threat. Finally the forth layer shows the preventive strategies. According to the system, some of the strategies in dependability diagram would contradict each other. For example using firewalls to improve safety may lead to availability deduction. The high level policies should confirm the proper trade offs [15].

To describe such diagrams, these points should be noticed:

- The system should be completely analyzed to find the main dependability attributes.
- The vulnerability and threats for each attribute should be determined.

- According to each threat, the appropriate treatment should be advised (preventive or tolerating).

Having such diagram, each low level component should be only equipped with appropriate information about the initial goals. So there is no need to have identical shared language. The vendors should prepare special strategies in the products in their desired formats. But these strategies should follow the concepts of the initial goals. Fig. 8 shows an autonomic computer system with goal-based translation strategy. The balloons are the resulted data of each engine. The system has three layers. The knowledge layer contains three engines: Monitor, analyze and plan. In this scenario, the monitoring engine detects attack situation and sends the state information to the analyzing engine. This engine infers the characteristics of the attack and then asks the planning engine to perform the appropriate actions. The planning engine prefers system to increase the overall security level. This decision is transmitted to all of the low level components. A hard disk drive is one of these components. The HDD execute engine tries to accomplish the decision by HDD capabilities such as creating extra copies of the critical data and data encryption. Other peripherals also carry out the appropriate actions depending on their structure and their dependability diagram. The peripheral effectors then finalize the process. Each engine is collaborating with the knowledge base during the process.

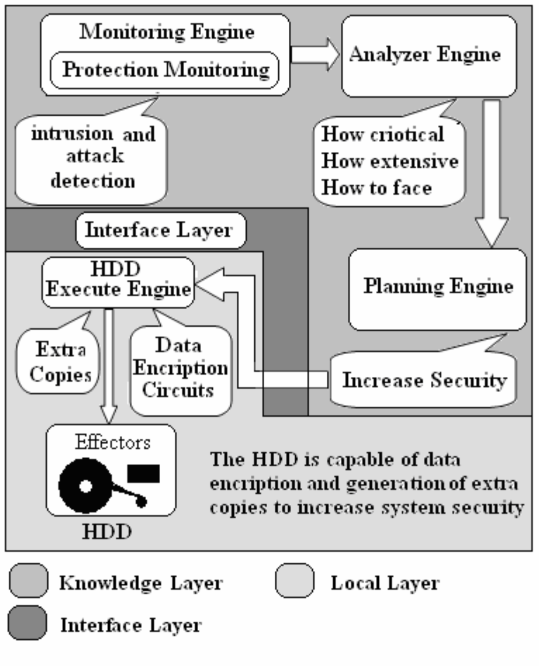

Fig. 8 An autonomic computer system using goal-based strategy

According to this scenario, it is evident that goal-based translation is not a revolutionary process and it can be though as an evolutionary strategy. This evolutionary nature is one of the best advantages of the strategy, since it is not needed to change the current infrastructures. For example, most of current hard disk drives have built-in data encryption feature, and they need only a light context translation support.

5. Goal-Based Translation and Future Interwoven Systems

The increasing complexity level of computer systems will lead to huge interwoven system of systems that are more heterogeneous and distributed than current systems [1]. These systems will need intelligent managements and supports that can not be achieved by today management systems. This kind of management will be based on semantic models to support scalability and fast decision makings. Since goal-based strategy is based on semantics and context of the autonomic goals, it would be an appropriate path to achieve the fully automated environments. Power of this strategy is based on the knowledge base behind it. Because of its dynamic nature, the knowledge base is dynamically improved making this strategy a learning model and a good infrastructure to support the future interwoven systems.

6. Conclusion

Goal-base translation strategy is based on a new view of the autonomic systems that tries to answer the complicated challenges of the heterogeneous environment and interwoven systems and adapting current infrastructures for future autonomic environments. The three layered model of this strategy supports the high level system policies and low-level system attributes and capabilities to achieve the final goal or self management. This strategy will make an independent intelligent knowledge cloud infrastructure for future distributed systems. On the other hand, using a dynamic knowledge base, an adapting interface layer and semantic based negotiations can make learning, scalable and adapting environment. Such evolutionary environments are critical requirements for the future generation of computing interwoven systems.

References:

[1] K. Hermann, G. Muhl, K. Geihs, Self Management: The solution to complexity or just another problem, *IEEE Distributed systems Online,*Vol.6, No.1, 2005.

[2] A. Ganik, T. Corbi, The dawning of the autonomic computing era, *IBM systems journal,*Vol.42, No. 1, 2003, pp.5-18.

[3] D. Chess, A. Segal, I. Whalley, S. White, Unity: experiences with a prototype autonomic computing system, *Proc.first IEEE Conf. on Autonomic Computing,* New York, NY, 2004, pp.140-147.

[4] G. Kaiser, J. Parekh, P. Gross, G. Valetto, Kinesthetics eXtreme:an external infrastructure for monitoring distributed legacy systems, *Proc. 5th Annual International Workshop on Active Middleware Services,* Seattle, WA, 2003, pp.22-31.

[5] S.W. Cheng, A. Huang, D. Garlan, B. Schmerl, P. Steenkiste, Rainbow architecture-based self adaptation with reusable infrastructure, *IEEE computer,* Vol 37, No.10, 2004, pp.46-54.

[6] H.Yan, D. Garlan, B. Schmerl, J. Aldrich, R. Kazman, DiscoTect : a system for discovering architectures from running systems, *Proc. 26th ACM/IEEE international Conf. on Software Engineering* .Edinburgh, Scotland, 2004, pp.470-479.

[7] D.A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, N. Treuhaft, Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175,* 2002.

[8] A. Lapouchinan, S. Liaskos, J. Mylopoulos, Y. Yu, Towards requirements-Driven autonomic systems design, *Proc. design and evolution of autonomic application sotware* St.Louis, MO, 2005, pp.45-51.

[9] K. Birman, R. Van Renesse, J. Kaufman, W. Vogels, Navigating in the storm: using astrolabe for distributed self configuration, monitoring and adaptation, *Proc. 5th Annual International Workshop on Active Middleware Services,* Seattle, WA, 2003, pp.4-13.

[10] H.A. Muller, L. O'Brien, M. Klein, B. Wood, *Autonomic computing* (Software Architecture Technology; Carnegie Mellon University, 2006).

[11] M. Agarwal, et al. , Automate: enabling autonomic applications on the grid, *Proc. 5th Autonomic Computing Workshop, Annual International Workshop on Active Middleware Services*, Seattle, WA, 2003, pp.48-59.

[12] NCW Roadmap, (Defence publishing center; department of defence, Canberra, ACT, 2005).

[13] M. Amin, Towards **s**elf-healing infrastructure systems, *IEEE. Computer Magazine V*ol.33, No.8, 2000, pp.44–53.

[14] T. Gruber, A translation approach to portable ontology specifications, *Knowledge acquisition* Vol.5, No.2, 1993, pp.199-220.

[15] R. Sterritt, M. Parashar, H. Tianfield, R. Unland, A concise introduction to autonomic computing, *Advanced engineering informatics* Vol.19, 2005, pp.181-187

[16] A. Avizienis, J. C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, *IEEE Transactions on Dependable and Secure Computing,* Vol.1, No.1, 2004, pp.11-33.