Towards An Automatic IT Infrastructure Management Paradigm for Manufactory and Enterprise

Jie Wang
Stanford University, CA, U.S.A.

ABSTRACT: Mission critical manufactory and enterprise IT infrastructure and on-line Internet services require 24x7 availability and usually have an agreed Service Level Agreements (SLA) around its services offering. Currently, most enterprise and manufactory IT datacenters experience severe problems managing the large N tier, networked environments that these mission critical systems run on. Most of these problems arise from the complexity of the infrastructure (containing many moving parts) itself rather than building the applications. Maintainability of the IT system for long-term operational efficiency is largely missing. On the other hand, new IT technologies aiming at automating the deployment and maintenance of IT infrastructure are emerging. One promising technology is that of autonomic computing where the IT infrastructure and its components are self-configuring, self-healing, self-optimizing and self-protecting and thus free of the management complexity to a large degree.

This paper proposes a management model for enterprise IT infrastructure using self-managing paradigm for enterprise IT system and then describes the procedure for using the model as the basis for providing realistic Quality of Service (QoS) and SLA for IT infrastructure services. We also suggest this autonomic infrastructure management model be applied to the management of complex manufacturing systems, including the management of the enhanced agile/lean/integration and configurable manufacturing systems.

Key Words: autonomic computing enterprise IT IT infrastructure topology based model SLA QoS IT management paradigm agile manufacturing

1 INTRODUCTION

1.1 Complex IT Systems and Infrastructure Management

While huge advances has been made in simplifying the development on distributed enterprise IT infrastructures, managing these infrastructures still remains a daunting task. The lack of management tools in this space coupled with the increasing dependence on the availability of these systems is causing significant resources to be put into managing these infrastructures once developed and there is a pressing need to address the space of IT infrastructure management and QoS.

Complexity has been the number one issue for managing a large enterprise IT infrastructure [6]. Across all industries, more than 40% of all investments in information technology are used just trying to get technologies to work together. In other words, almost half the investment goes for things that don't directly drive business value.

1.2 Managing IT and its Infrastructure Systems

One of the most urgent things for managing enterprise and manufactory IT is to know the IT system itself. Based on this information, we can then discuss the possibility of a more automatic management paradigm. The one of the fundamental steps for achieving the ambitious goal is to find the model that can describe the IT system. Once we establish a proper model, we can use it for

1. monitoring the real-time system,
2. better trouble shooting the system,
3. estimating the reliability of the system, and

4. calculating the QoS and SLA based on the reliability.

This paper focuses on the issues for establishing the IT infrastructure model and for estimating the reliability of the infrastructure system. In addition, we will discuss the methodology for building up rational QoS and SLA based on the reliability of the underlying systems and infrastructures.

2 SELF-MANAGEMENT PARADIGM FOR IT INFRASTRUCTURE

2.1 Current Research Topics

The IBM autonomic computing manifesto presents us a new hope, as well as a grand challenge, to develop and deploy systems and software that can run themselves, can adapt to varying circumstances and can operate robustly under the damaged conditions and degradation. In reality, IBM's vision is an amalgam of analogies from biology. The primary analogy is to self-regulating control systems that maintain steady state. It also draws on ideas that relates to the immune system. While research for these analogies begin to draw more attentions [9], building software that is aware of its own behavior, its surrounding context, and the control and interaction among components is extremely difficult.

To address the challenges, current research on autonomic computing has been focused on the following several major areas:

(1) **Autonomic computing techniques and self-healing systems architecture: using principles learned from biologic system, adaptive system, etc for designing autonomic systems.** The rational for using biological system shows as follow: since human efforts to design and engineer self-healing systems prove limited success, some researchers [8] argue that the concept and process in the broader biological cell programs the and strategies they use to robustly accomplish complex tasks such as development, healing and regeneration should be investigated for building new model for autonomic computing environment. The second research topic in this area focuses on reflection and self-awareness in self-healing systems and the Meta model for integrating and controlling the system components [2]. The third research topic studies the over-all architecture and architecture requirement for a self-healing system [7,10].

(2) **Multiagent systems for automatic detection and self-organization**. Using multiagent technique and the related artificial intelligence (AI) principles for building autonomic systems [1]. The focus there is to demonstrate that by building a general purpose multiagent system for autonomic system environment, one is able to better deal with software system management, including event processing, performance monitoring using adaptive thresholds, system health monitoring using hierarchies of fuzzy rules, and time-series prediction for service-level agreement management using neural networks. The next step for the research is to add the support for integrated method for encoding and reasoning about IT infrastructure using information modeling and representation, knowledge inference and sharing.

(3) **Recovery-Oriented Computing (ROC) [5,11] and software rejuvenation [13]**. Research in those areas focus on the modeling of system failures and the most feasible and economical way for a recovery from the failures in a real scenario IT infrastructure operational environment. While ROC concentrating on Mean Time to Repair (MTTR) for a system recovery, software rejuvenation approach focuses on a Markov process model for inspection based preventive maintenance in operational software systems. These researches are helpful for understanding the fast recovery for a current It infrastructure, but don't provide fundamental methodology for dealing with the mechanisms for architecting autonomic computing..

(4) **New approaches for automatic IT infrastructure management [12]**. Improving the current infrastructure software management tools and bringing them to a level that the tools can organize the IT infrastructure operation around the business service need and IT users can monitor the business processes, not the complex of the IT system itself. While most researches in this area are still focusing on the theoretical and experimental aspects, the automatic IT infrastructure management research has set up a goal for business service purposes and commercial development.

3 TOPOLOGY BASED MODELING FOR IT INFRASTRUCTURE

The starting point for self-managing for IT system is to know oneself and so determining the topology of the IT applications first defines a natural progression. One cannot deploy an IT application without having an idea of the different static parts of it and how they are packaged. One cannot automatically monitor an IT system not having the dynamic picture of the application components and any root cause isolation needs the relationships between components in order to be accurate. To this purpose, we therefore define topology as the combination of the static description of the application's components and the relationships that exist between these components. All applications have 2 parts to them: (1) Infrastructure, and (2) Application entities. And the model for IT infrastructure needs to include entities for the above two parts and their relations.

More specifically, a model for infrastructure can in turn be considered to consist of 3 tiers as follows: (1) Network Tier: This consists of devices such as switches, routers and load balancers along with firewalls that divide the network into multiple subnets and “firewall zones”. (2) Systems Tier: The systems tier provides the computing infrastructure for the software and consists of the compute servers along with the operating system. (3) Applications Infrastructure Tier: This tier provides the software “containers” in which application components execute. We can think of any number of container types to house the corresponding type of application entity. For example: a data base container will house application schemas, stored procedures, a web container will host content and dynamic components etc.

Modeling application entities should take into account the following considerations. Most distributed multi-tier applications also have a well-defined architectural structure to them. They consist of thin clients attaching to computing tiers that are sets of cooperating processes accessing a set of DB processes at the back end. The more interesting parts of the application are the "middle" tiers and the backend since, by definition; thin clients don't have the meat to them to bear management. There is yet another angle to this – an application or a set of cooperating applications offer “services” to its users. A service can be considered to be a set of related transactions. Examples can include web services such as that of ordering on an eCommerce site.

4 ESTIMATING THE RELIABILITY OF IT APPLICATIONS AND INFRASTRUCTURES

The reliability of an IT infrastructure can be estimated based on the underlying infrastructure model. As a first step, we should establish a reliability model for the IT infrastructure. The model reflects the following intuitive system feature:

- In general, the reliability of a system decreases as the complexity of the system increase.
- If there is a single point of failure in the system, the system cannot be more stable than that component.
- Redundancy enhances reliability.
- However, redundant components may be prone to fail at the same time because the failure may have the same cause.

The reliability estimation method includes the following four steps, which can be summarized as:

- Mapping the system as a graph.
- Determining the critical graph for outage.
- Assessing the reliability of basic modules.
- Computing the system reliability.

5 USING RELIABILITY MODEL FOR QOS AND SLA

Corporate IT infrastructures have become key repositories of business information needed by employees and clients across the enterprise and through the Internet. Companies also rely on the existence of network-based IT services for their businesses, running mission critical applications for ERP, CRM, eCommerce, and more. Quality of service (QoS) solutions can help enterprises cope with the unprecedented demands on management of their IT infrastructures.

There are basically two approaches to deliver QoS. One is to create business service requirements that need a very high IT infrastructure reliability. For this approach, we calculate the current reliability of the infrastructure and if it cannot support the business requirements, we need to make a predication, which is based on simulations of reliability model for planned infrastructures, for an acceptable IT infrastructure to fulfill the requirements. The second approach is to dispense with the current IT infrastructure, based on the calculation of the best capability and usage of the infrastructure using the reliability model, to prioritize the business requirements and maximize the business service offering. Both approaches need the information of the reliability model for the IT infrastructure. In summary, we are able to estimate QoS based on the reliability model for IT infrastructure.

After we understand the available QoS for the infrastructure, we can establish SLA for the IT service. SLA monitoring and enforcement become increasingly important in an IT service environment. By using the predications obtained from a reliability model and the estimated QoS, we can set up the foundation for supporting SLA of the service offerings. By using SLA, enterprise IT applications and services may be subscribed dynamically and on-demand based on the business requirements and priorities.

As a result, for economic and practical reasons, we should use the model based IT management paradigm to arrive at an automated process for both the service itself as well as the SLA management system that measures and monitors the QoS parameters, checks the agreed-upon service levels, and reports violations to the authorized parties involved in the SLA management process.

6 SUMMARY

In summary, the field of self-managing systems is only now coming of age and distributed systems such as those built on middleware tend to be complex and are not easy to manage. We present a topology based model for modeling IT infrastructure. We believe that a formal model for IT systems is the first step of being able to build self managing systems.

Based on the model, a better self-management of the IT systems can be achieved. From IT service view point, the model offers the basis for estimating QoS and SLA for the service offerings. From economical view point, the model provides a foundation for mitigating the business losses due to infrastructure failures.

We believe that the proposed infrastructure systems management paradigm can be applied to the management of other complex manufacturing systems such as the enhanced agile/lean/integration and configurable manufacturing systems. We hope that through the collaboration with experts and professionals in industrial and manufacturing systems engineering domains, we will be able to employ this methodology to improve the quality, reliability, and sustainability of complex manufacturing systems.

REFERENCES

[1] Bigus, J. P., et al., "A Toolkit for Building Multiagent AutonomicSystems", http://www.research.ibm.com/journal/sj/413/bigus.html, 2002.

[2] Blair, G., et al., "Reflection, Self-Awareness and Self-Healing in OpenORB", ACM WOSS, Charleston, SC , 9-14, Nov., 2002.

[3] Dabrowshi, C. and Mills, K., "Understanding Self-healing in Service-Discovery Systems", ACM WOSS, Charleston, SC, USA., 15-20, Nov., 2002.

[4] Dashofy E. M.m et al., "Towards Architecture-based Self-Healing Systems", ACM WOSS, Charleston, SC, USA., 21-26, Nov., 2002.

[5] Fox A. and Patterson, D., "When Does Fast Recovery Trump High Reliability?", Proceedings of the EASY 2002, San Jose, CA, October 2002.

[6] Ganek, A., "A letter from Vice President, Autononomic Computing, Alan Ganek", http://www-3.ibm.com/autonomic/letter.shtml, 2002.

[7] Garlan, D. and Schmerl, B., "Model-based Adaptation for Self-Healing Systems", ACM WOSS, Charleston, SC, USA., 27-32, Nov., 2002.

[8] George S., et al., "A Biologically Inspired Programming Model for Self-Healing Systems", ACM WOSS, Charleston, SC, USA., 102-104, Nov., 2002.

[9] IBM paper-1: "IBM autonomic computing challenges note: academic focus article: challenges", http://www.research.ibm.com/autonomic/academic/challenges.html, 2002.

[10] Mikic-Rakic, M., et al., "Architectural Style Requirements for Self-Healing Systems", ACM WOSS, Charleston, SC, USA., 49-54, Nov., 2002.

[11] Patterson, D., et al., "Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies", In Proceedings of the UC Berkeley Computer Science Technical Report UCB/CSD-02-1175, Berkeley, CA, March 2002.

[12] Tivoli software, "Autonomic Computing: The Value of Self Managing Systems", http://www.tivoli.com/news/features/oct2002/autonomic.html, 2002.

[13] Vaidyanathan, K., Selvamuthu, D., and Trivedi, K. S., "Analysis of Inspection-Based Preventive Maintenance in Operational Software Systems", Intl. Symposium on Reliable Distributed Systems, SRDS 2002, Osaka, Japan, October 2002.