

The Role of Agents in Enterprise System Management: A Position Paper

Onn Shehory
IBM Haifa Research Lab
Mount Carmel, Haifa
31905 Israel
+972-7689462
onn@il.ibm.com

ABSTRACT

The size and complexity of enterprise computer systems are growing rapidly. As a result, system management has become increasingly difficult and expensive. In fact, management costs are typically estimated at 50%-70% of the total cost of ownership. Despite large investments in management software and personnel, enterprise computer systems are usually managed sub-optimally. This situation calls for a fundamental change in the way systems are managed.

Recent studies suggest that systems manage themselves autonomously. Initial studies towards that goal reside in the fields of Autonomic Computing and Self* Systems. Many of those studies devise mechanisms for system monitoring, data filtering, problem determination, problem fixing and system adjustment and reconfiguration. The mechanisms are based, in large part, on methods from Artificial Intelligence, Statistics, Operations Research and Software Engineering.

Current enterprise system management solutions – theoretical and practical – are centralized. However, the typical enterprise system is distributed. We claim that a centralized solution presents several risks. For instance, it introduces a management bottleneck, it incurs communication overheads, etc. We hence advocate that enterprise system management should be addressed in a distributed manner. Combining distribution and AI, we believe that an agent-based solution is appropriate. In this paper we present in more detail the suggested approach.

Categories and Subject Descriptors

C.4 [Performance of Systems]

I.2.11 [Distributed Artificial Intelligence]

General Terms

Management, Performance, Reliability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

1. INTRODUCTION

The size and complexity of enterprise computer systems are growing rapidly. Contemporary enterprise systems typically include thousands of hardware and software components distributed across networks and geographies. These components are heterogeneous in their functionality, their interface with other components and with human users, and their manufacturers. Additionally, management practices and tools, when provided with the components, vary considerably in their capabilities. For instance, some tools merely monitor a small set of operational parameters and provide a report on their values, whereas others may be collecting a large and diverse set of parameters and provide complex analysis such as trending, prediction and bottleneck analysis. Note however that, even had all of the components been provided with the best management tool each, the management of the entire enterprise system will not instantaneously be resolved.

In managing an enterprise computer system, one needs to manage not only its components, but the relationships between them as well. Theoretically, this introduces a combinatorial problem, as any combination of components should be considered in the solution. Indeed, in recent years system management has become increasingly difficult and expensive. The costs of management are typically estimated at 50%-70% of the total cost of ownership, as can be inferred from multiple market studies (see e.g. [12]). Large investments in management software have not yet produced good solutions. Apparently, as many system administrators admit, enterprise computer systems are usually managed sub-optimally. This situation calls for a fundamental change in the way systems are managed. We advocate that this change will require a combination of Distributed Systems [9] and Artificial Intelligence techniques. The discipline that combines such techniques is Distributed AI—the parent discipline of agents and multi-agent systems.

Recent studies suggest that systems manage themselves autonomously [5]. Initial studies towards that goal reside in the fields of Autonomic Computing [3][6] and Self-* Systems [8]. The idea of systems that manage themselves is not new, though. Computer scientists have envisioned such systems since the era of Turing and von-Neumann [10][11]. Attention to these ideas has returned only in the last few years. Research in the area studies, among others, self-diagnosis, self-organization, self-configuration, self-awareness, self-management, etc. Many of these studies

devise mechanisms for system monitoring, data filtering, problem determination, problem fixing and system adjustment and reconfiguration (e.g., [2]). The mechanisms are based, in large part, on methods from Artificial Intelligence, Statistics, Operations Research and Software Engineering. Notably however, techniques for addressing distribution-related other than networking aspects of self-management are rarely employed. Further, the extent to which solutions address heterogeneity is limited.

Current enterprise system management solutions – theoretical and practical – are mostly centralized, although the typical enterprise system is distributed. Management solutions require human intervention and they are far from being autonomous. We argue that a centralized solution presents several limitations and risks. For instance, it introduces a management bottleneck, it incurs communication overheads, etc. We hence advocate that enterprise system management should be addressed in a distributed manner. We also argue that increased autonomy of the management system should reduce its TCO. As suggested above, we believe that an agent-based solution is appropriate. In this paper we present in more detail the suggested approach.

2. CURRENT SOLUTIONS

The goal of an enterprise computer system is to provide the enterprise with services of data processing, data communication and data storage. These services should be provided at a certain quality, to support the business operations and goals of the organization. The goal of system management is to assure that the system indeed provides the services expected, at the quality expected.

For instance, consider an organization whose business is graphical design. One goal of the system may be the retrieval of graphical files at a maximum delay of one second, regardless of the location of the file and its requestor. Of course, one should expect the initial system configuration to support this requirement. However, since data volumes and usage patterns change over time, and since system components may fail or degrade, the system might, in times, fail to address the required retrieval time. The major means for reducing the rate of occurrence of such problems is high-quality system management.

As stated above, the ultimate goal of system management is to assure service provision at a certain quality. Had management solutions fully addressed this goal, there would be no other requirement from these solutions. However, since no solution fully addresses the management goal, other requirements are typically set. These include the following:

- Reports on the system status: are the goals set for the system satisfied? What are the deviations from these goals?
- Trending and prediction: what are the system's patterns of operation? What system status would they lead to?
- System analysis, e.g.:
 - Dependency analysis: which application or component depends on which, and in what ways?
 - Root-cause analysis: what is the root-cause of a reported problem?

- Bottleneck analysis: which components in the system experience the highest load, and which ones will fail first as a result of a high load?

- Remediation: what can be reconfigured or fixed, and how, to resolve reported problems, and to avoid problem build-up?

The requirements listed above are addressed to a limited extent by existing solutions. Fielded solutions, and in particular enterprise system management tools, mainly address the reporting and trending requirements. That is, such tools perform the following:

- Monitor the system – its components and applications – and store the collected data (possibly after some filtering and processing).
- Generate reports from the stored data
- Detect abnormal system and component status
- Compute simple trends
- Support a restricted set of component and system reconfiguration actions, including automated component replacement

The management functions listed above are not supported by all of the management tools. In practice, many tools offer a rather limited management function: they usually monitor the system and report on system status and trends, detect some types of abnormal behaviour, and, in some cases, they allow for (partial) automated reconfiguration of the system to address detected problems, commonly in a human-in-the-loop manner.

Solutions to the system management problem performed in research labs and published in the literature provide a variety of functions which are not yet part of fielded solutions. In particular, analyses such as dependency analysis, root-cause analysis, bottleneck analysis and prediction are addressed (e.g., [1][4][7]). Additionally, system re-configuration and a few self-healing functions are also resolved. However, research-grade solutions have several limitations, as follows:

- They usually solve a single, isolated problem, and do not provide a comprehensive solution as a management tool should.
- They tend to be theoretical, in the sense that they typically make assumptions that do not hold in the field.
- Commonly, they do not attempt to reduce the computational complexity of the suggested solution

Despite these limitations, it is expected that research solutions will eventually move into fielded tools. Note however that research solutions, although more advanced than tools, are far from addressing all of the requirements listed above. This invites additional research, and as stated above, agents and multi-agent systems could be the right discipline for this research.

3. TYPICAL ARCHITECTURES

In this section we describe the common architectural properties of enterprise system management tools. Although typical enterprise systems are distributed systems, both the fielded and the research solutions to the management problem are mostly centralized.

A management tool consists of five major layers: monitoring infrastructure, a database, an analysis layer, an activation layer and a user interface. An illustration of a typical architecture is depicted in Figure 1, where the management tool appears in grey. Typically, the monitoring infrastructure is not monolithic. Rather, it is comprised from multiple monitoring elements. Each such element (in times referred to as an agent) may be located on, or near, the component(s) it monitors. It collects data from the components it is assigned to using standard protocols such as SNMP and CIM/WEBM. The monitoring elements send the collected data, in some cases after some simple filtering, to a *central* management point to be stored in the database and analyzed as found appropriate.

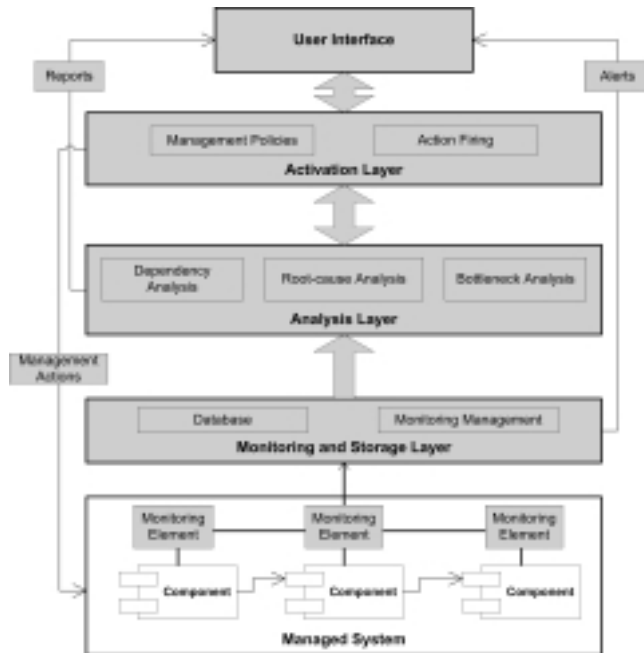


Figure 1. Management tools: a typical architecture

The analysis layer is typically located at the central management point. It may be implemented as a single monolithic software component, or as a collection of such components each specialized in a specific type of analysis. The analysis layer retrieves from the database data collected and filtered by the monitoring layer. The results of the analysis are sent either to the user interface as reports, or to the activation layer, for policy updates, planning and action firing.

The activation layer, located at the central management point as well, consists of an action firing component. It usually consists of means for representing management policies and activation rules too. Action firing is performed either by the user, or automatically via activation rules, following management policies. Actions are performed directly on system components and may include parameter setting, component and system reconfiguration, component replacement, and others.

The user interface usually runs at the central management point, however web access to the user interface is becoming increasingly common. The user interface provides the user with access to an array of management functions. These include systems visualization, reporting and trending, analysis activation and

results, action firing (see details above), and management of the management tool itself.

The architecture described here does not cover all types of management tools, however it represents well many of them. One of the prominent common properties of these tools is their centralization. Typically, such tools will be running on a server dedicated to system management. The advantages of this approach are the computational efficiency of data retrieval and analysis, and the high reliability of the servers on which the management software runs. However, the centralized solution has several weaknesses, as follows:

- **Communication overloads:** Frequent monitoring of multiple components across the managed system generates large data streams. These are sent to the central management server, possibly overloading the server and the network. Contemporary centralized management tools address this problem by reducing the frequency of their monitoring. As a result, the quality of management may be compromised.
- **Computation overloads:** When multiple analysis methods are activated concurrently to analyze a large number of components and their operational parameters, a central server might exhibit computation overloads, which will in turn result in slow analysis and delayed response to problems detected in the managed system.
- **Single point of failure:** A widely known problem of centralized solutions is them being a single point of failure. In this respect, system management tools are not an exception. A failure of the management tool may result in degradation in the quality of services provided by the managed system, and may lead to its failure.

The weaknesses discussed above call for a change in the way management tools are designed and deployed. In particular, distribution could provide some remedy to all issues raised. However, distribution may introduce new problems, as follows:

- **Synchronization and consistency:** Once the management of a system is distributed, the widely known clock synchronization problem arises. Additionally, when data is distributed across multiple management points, inconsistency among them may arise. These problems may in turn result in errors in data analysis and reporting, and in action firing.
- **Communication overhead:** The distribution of management across the managed system requires coordination among the managing points. This in turn results in increased levels of communication traffic and may impose a communication overload. Careful design of the coordination can minimize this risk.

It appears that, although distribution could provide significant advantages to system management tools, there is a need for a careful design to reduce the risks introduced by distribution. In the following we propose a possible approach to the distributed design of enterprise system management tools.

4. AN AGENT-BASED APPROACH

Given the requirements for data collection, filtering and analysis, and for action firing, across a distributed heterogeneous system,

we believe that an agent-based solution is adequate. Our proposed solution approach is as follows:

- **Data collection:** In current solutions, data is collected and sent to a central management point. We suggest that collection will be performed by agents that implement some intelligence. In particular, they should intelligently filter the data collected, and transmit, either to a central point or to other management entities, a small subset of the collected raw data. This should resolve the communication overhead present in current solutions.
- **Data analysis:** In current solutions, analysis is performed at a single central point. We suggest that each type of analysis be performed by a specialized component or agent, possibly located on a separate machine. This should resolve the computation overhead present in current solutions.
- **Action firing:** In current solutions, management policies are maintained at, and actions are fired from, a single central point. We suggest that some policy maintenance and action firing will be moved to the agents residing next to the managed components. In particular, some policies and actions are relevant only to a small subset of the system, and it should be efficient to maintain the former, and fire the latter, in vicinity to their target components. By this, communication overhead will be reduced and timeliness of management actions will be improved.
- **Coordination and collaboration:** In current solutions, due to their central approach, no coordination or collaboration are implemented. In a distributed solution where data collection, filtering and analysis, and action firing, are performed locally by agents, there is a need to coordinate among them to avoid inconsistencies and prevent conflicting and redundant actions. Collaboration could further enhance the capabilities of the management tool. We suggest leveraging multi-agent coordination and collaboration methods to address this goal.

5. CONCLUSION

The solution approach suggested in this paper is described at a conceptual level. The main aim of this description is not to provide guidelines for developing agent-based management systems. Rather, we want to raise community awareness to a problem domain in which, to date, little was studied, and where agent-based solutions appear as a promising solution approach. As we demonstrate, distribution, local autonomy, intelligence and coordination could provide significant improvements to the quality of enterprise system management solutions. Since agent-based systems inherently realize these properties, we believe that

agents and multi-agent systems should be studied and utilized in this context. We are confident that the results of such research will span far beyond this specific application domain, influencing the disciplines of system management, self-*, autonomic computing and, of course, agents and multi-agent systems.

6. REFERENCES

- [1] Breitgand, D. Henis, E. Ladan-Mozes, E. and Yerushalmi, E. Root-Cause Analysis of SAN Performance Problems: an I/O Path Affine Search Approach. In IFIP/IEEE 9th International Symposium on Integrated Network Management, IM'05, Nice, France, May 2005.
- [2] Dilman, M. and Raz, D. Efficient Reactive Monitoring. IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Recent Advances in Network Management, 20(4).
- [3] Ganek, A.G. and Corbi, T.A. The Dawn of the Autonomic Computing Era, IBM Systems Journal, Vol. 42(1), 2003.
- [4] Hellerstein, J., Zhang, F. and Shahabuddin, P. A Statistical Approach to Predictive Detection. IEEE Computer Networks, January 2000.
- [5] Herrmann, K., Mühl, G. and Geihs, K. Self-Management: The Solution to Complexity or Just Another Problem?. IEEE Distributed Systems Online, V 6(1)1, 2005.
- [6] Kephart, J., Chess, D. The vision of autonomic computing. IEEE Computer, January 2003, 41-50.
- [7] Kar, G., Kochut, A., Bobroff, N. and Beaty, K. Management Issues in Storage Management Networks: Detection and Isolation of Performance Problems. In IFIP/IEEE 9th International Network Operation and Management Symposium (NOMS'04), Seoul, Korea, March 2004.
- [8] Proc. of SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems, 31 May - 2 June 2004, University of Bologna, <http://www.cs.unibo.it/self-star/program.html>
- [9] Tanenbaum, A.S. and van Stee, M. Distributed Systems: Principles and Paradigms, Prentice-Hall, 2002.
- [10] Turing, A.M. Intelligent Machinery, 1948, Report for National Physical Laboratory, in Machine Intelligence 7, Meltzer, B. and Michie, D., Eds. 1969.
- [11] von-Newmann, J. and Burks, A.W., Eds. Theory of self-Reproducing Automata, University of Illinois Press, 1966.
- [12] <http://www-1.ibm.com/linux/whitepapers/robertFrancesGroupLinuxTCOAnalysis05.pdf>