Architecting Principles for Self-Managing Enterprise IT Systems

Kemal A. Delic, Jeff A. Riley, Yassine Faihe
Hewlett-Packard Comp.
name.surname@hp.com

Abstract

Economic mega-shifts and technology advances have created an entirely new context for the enterprise IT systems. It is now a very common expectation that the cost/performance ratios will be constantly improving while always offering better services and novel IT features. To fit such expectations, enterprise IT systems should be architected differently. In general, cost savings should be used to finance innovative projects fitting into enterprise architecting blueprints. In this paper we give a top-level view into typical enterprise IT system and outline four architecting principles to guide the implementation of these innovative projects articulated together as 'renewal program' across entire IT. After describing in more depth each principle, we conclude with some practical challenges and outline a few interesting research directions. We also share some practical insights from an internal HP project.

1. Introduction

Business enterprises need information and communication infrastructure (ICT) to operate in markets and run efficient and profitable business. We consider any business with more than 5000 employees and with revenues surpassing a couple of billion dollars as an enterprise. Enterprises of such large size and scope are typically in business for several decades and have accumulated various types of technologies and applications which are considered as 'legacy sediments'.

Enterprise Architecture (EA) is a visionary blueprint which will ensure investments in IT so that the future business growth and shrinking operations are accommodated properly. Architecting an enterprise IT infrastructure means; devising a set of long-lasting, accurate principles which will support technical decisions and guide appropriate and timely investments. The overall effect, observed externally, is that the business passes renewal through creation of new products, services, entering into new markets or through merger, acquisition or divestiture operations which is followed by IT renewal.

This renewal is based on two principal acts: (1) cost saving and (2) accurate IT investments. At the global level we observe a very important change: the typical spread of enterprise IT cost which consists in 80% on operations and 20% on innovation is now inversed. A successful renewal is marked by the rebalance of IT budget with 80% devoted to innovation. We argue that the enterprise IT renewal can be enabled by applying four architecting principles – simplification, virtualization, automation and aggregation – to innovative projects. Problem analysis will be the essential starting point for each practical application of these principles.

In this paper we describe each of the four architecting principles. We are outlining "what' without detailing "how", as ordered application of given principles (simplification-> virtualization-> automation-> aggregation) represent a high-level IT renewal program by itself.

2. Simplification: Aim to deal with complexity

The long-living enterprises exhibit some complex behaviors as they consist of great many components (thus, they are 'complicated') which interact with people (principal source of uncertainty) in a not-always predictable (or process prescribed) manner. Dealing with them is more an art today than an established science.

Consider, for example, recent infrastructure of HP enabling several core businesses to operate smoothly across the globe (Fig1). Highly simplified picture can be used by the corporate executives to establish appropriate context and get sense of the overall corporate IT layout.

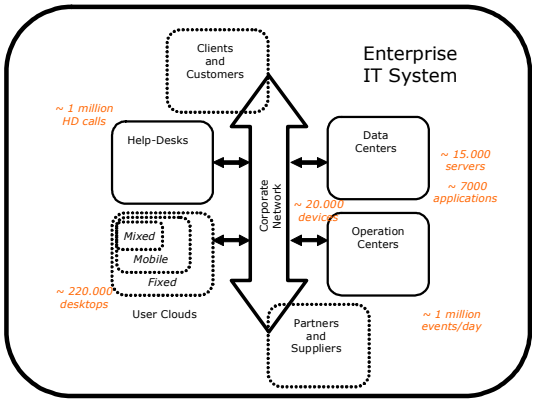

Fig. 1. Enterprise IT System – Conceptual View

Having understood the IT layout and its importance for the links with clients and customers while being supported by partners and suppliers, executives will typically ask for some key operational figures and ratios in order to understand where to apply 'the principle of simplification'. There are many ways to improve enterprise IT and to apply 'simplification' on several starting points and levels. Thus, limiting variations through standardization could be a way to start. This will reduce support cost and focus IT personnel on few crucial skills. It will also enable concentrated effort on harvesting operational maintenance and problem-solving knowledge.

CxO type of executives typically look into telling ratios as servers/head, calls/analyst, events/device, GFlops/Kwh etc as indicators of cost/performance improvements. Unfortunately, they are not standardized, thus often are treated as 'gut feeling' or 'anecdotal evidence'.

3. Virtualization: Better use of resources

Once variety of IT components is reduced, the typical next step is to monitor and measure usage of enterprise ICT resources. Quite often, one will find that resources are heavily underused: typically only 20% of full capacity is being used. Thus, virtualization of resources is done on all levels: starting from the better use of single machine to virtualization of the huge data center's resources. In most publicly known, typical cases, we may expect usage to grow up to 70-80% of the full capacity.

Virtualization refers to a technique to simplify IT environments. It consists in aggregating computing resources by introducing a virtual layer to uniformly access to resources, and the same time allowing those resources to be shared across concurrent applications.

Virtualization is a key driver to server consolidation initiatives. In fact the technique has been used to create several virtual machines by using memory, CPU and I/O from a single physical machine. In addition to efficiently using the available pool of resources virtualizations allows to easily and rapidly provision additional resources should a service or application get expanded.

The impact of virtualization on IT maintenance is huge. First, with the decrease of the number of machines to be managed the cost of IT staff will also drop down. Second, since each application could potentially run on its own virtual machine and operating system, virtualization offers a perfect isolation from a performance, security and failure perspectives.

Utility computing paradigm is essentially enabled by virtualization, as we provide an environment in which IT resources are being exploited in innovative ways by several clients and customers. We would expect also the rise of IT utility markets in which utility services will be traded in a way similar to other established markets.

4. Automation: Changing Cost /Performance Ratios

Assuming that the simplification and virtualization have provided expected improvements, we focus here on automation as the principal source of cost savings. Therefore we describe in some depth the 'automation principle' as it applies to the critical maintenance area and in particular to problem detection and analysis (Fig 2).

The automation of system monitoring, problem recognition, analysis and, when necessary, routing to a human expert is a critical step in architecting self-managing systems (system able to emulate human managing chores). We describe here an intelligent system for the analysis of system state, problem recognition and routing. The system described employs a number of AI machine learning techniques to automate problem recognition and analysis. Problems not able to be diagnosed automatically are routed intelligently to human experts for further analysis and resolution, and the results of this manual step are fed back into the system, thus completing the adaptive diagnostic loop.

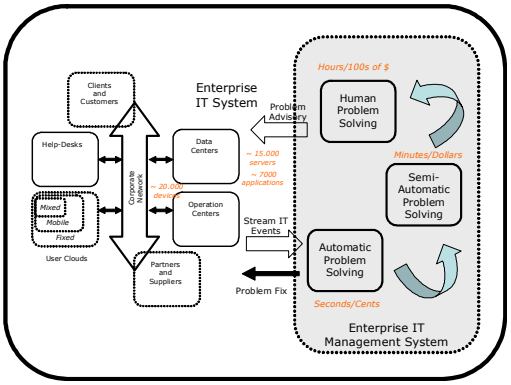

Fig. 2. Automation Principle Outlined

In this work, the system state at any time is described by the values of a predefined set of parameters (i.e. attribute-value pairs) relevant to representing normal (and abnormal) system operation. These parameters are monitored constantly and, at regular intervals, presented to the system state recognition engine for analysis. Choice of the training set will be critical for accurate functioning as different methodologies require either well-balanced set of examples or strongly skewed.

A. *System State Recognition*

A significant problem when dealing with large, complex, heterogeneous systems of any kind is how to recognize problem, or abnormal, situations and events. Humans do this largely through experience, but computer programs typically need to be trained on large amounts of data before they can do anything more than recognize specific events. Since there is likely to be more data available for "ordinary" or "normal" situations and events than there is for those that are out-of-the-ordinary or abnormal, it makes sense to train automated monitoring/recognition systems to recognize the normal operation of a complex system so that any situation or event not recognized is then considered abnormal and indicative that a problem has occurred (or is about to occur). This is the approach taken in this work.

If the current system state is recognized as abnormal or problematic, the system state recognition engine passes the problem to the known problem matching engine.

B. *Known (Previously Seen) Problem Matching*

Once an abnormal, or problem, state has been identified, that state needs to be compared with previously seen problem states to determine if a solution (or workaround) to the problem is known. Known problem matching is achieved by the use of categorizers that map the problematic system state to the appropriate solutions. Such a mapping is realized through one or more of the following AI machine learning techniques:

- A rule-based expert system [1] driven by expert domain design knowledge supplied by system designers.

- A comprehensive decision-tree analysis system [2] driven by expert domain operational knowledge supplied by system analysts.

- A machine learning technique such as neural network [3] constructed and trained by knowledge mined from historical problem resolution data.

The system state recognition engine will, by analysis of the parameters monitored, also provide the order in which the known problem recognition techniques should be executed. If any of the known problem matching techniques recognizes the current system state as a known, or previously seen problem, the problem solution or workaround details are retrieved from the relevant database.

If the system state is not recognized as a known problem, the problem is passed to the problem routing engine.

C. *Problem Routing and Adaptive Diagnostics*

If the known problem matching engine is not able to match the system state to any previously seen problems, the problem is routed to an appropriate human analyst. The known problem matching engine provides hints in the form of similarity likelihoods for various classes of problems (e.g. network administration, web server, application, database etc.). The engine matching capability can also be learnt using the same above-mentioned techniques. The predicted classes of problems are then used by the routing engine in conjunction with a skills database to determine an appropriate analyst or group to which the problem should be routed for further (manual) analysis.

After the problem is successfully routed to a human expert, the analyst will check known problem databases and use their own experience to determine if the problem can be solved readily. If the analyst finds a solution to the problem s/he will update the relevant data and knowledge bases so that the automated system is more likely to resolve the problem the next time it, or one similar, is encountered.

Problem resolution data learned by human experts as they analyze problems routed to them is fed back into the system so that future automatic analysis will have a better chance of recognizing and solving the problem. This is the continuous learning of the adaptive diagnostics loop.

D. *Offline Learning*

In order that the most complete and current information is used to drive the intelligent systems employed in this technique, the neural network described earlier in this section needs to be regularly trained with the most up-to-date problem resolution data. At regular intervals known problem databases are mined and problem recognition and resolution data are extracted and used to train the neural network. This offline learning is scheduled to run in the background during off-hours.

5. Aggregation: Reduction of support and operation cost

Aggregation may take different forms at various levels, while we point here at enterprise application as a generic enterprise application stack (similar to LAMP aggregation concept in open source world). It is intuitively clear that such an aggregation should reduce cost and support efforts significantly. Instead of having several experts for each layer (for SAN, database, application, web servers, firewalls and load-balancers) we aim to deal with higher level aggregate (Fig. 3.) which will be considered holistically, thus abstracting problems at the higher level of functionality.

Such type of enterprise applications represent the major workloads of enterprise data-centers and therefore, deserve careful in-depth analysis. Within aggregated EA stack, we should apply monitoring and control loops well defined within control systems theory, since we are indicating them here only in schematic way. This should be focus of further R&D work.

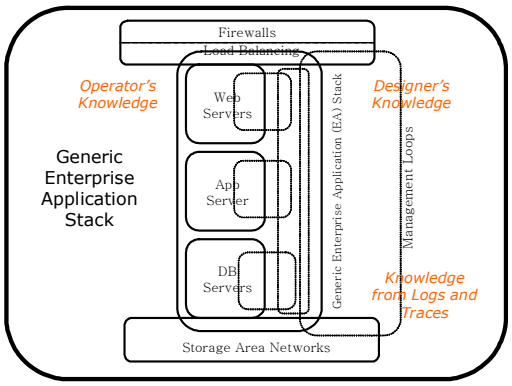

Fig. 3. Generic Enterprise Application Stack

Entire stack is managed through several embedded monitoring and control loops guided by knowledge components containing problem-solving knowledge for known set of problems. We point at the necessity of combining (at least) three sources of knowledge: designer, user/operator and knowledge from various logs and traces. As the nature of knowledge is different it is very likely that the knowledge representation will be different for different sources. Fusing of these knowledge types into coherent whole, remains to be a challenging research problem.

The next level of aggregation will be to integrate this software stack with appropriate hardware creating an 'enterprise appliance'. One should observe that automation principles described in section IV could be happily applied to appliance hardware and running enterprise software stack.

6. Outlook: Enterprise IT Renewal

Many enterprises have an in-house, self-grown IT infrastructure which is typically expensive, hard to manage difficult to evolve [4]. We have outlined a high-level architecting plan inspired by self-management in which we aim at evolving enterprise architecture exhibiting adaptive features. As the overall cost of enterprise IT goes between 7-15% of the revenues of which 80% is typically spent on daily operations, we postulate that the cost should be reduced on 3-4% and in 'renewed' IT, 80% should be spent on innovation.

In extension, cost reduction obtained should be ideally used for re-investments into more efficient technologies enabling thus enterprise IT renewal. This will stimulate innovative thoughts and solutions creating a yet another type of revenues from IT – Intellectual Property (IP). If we observe 10x improvements in performance, cost reduction, speed or effort, there is a strong chance that behind we have

highly invention solution. It should be captured and legally protected.

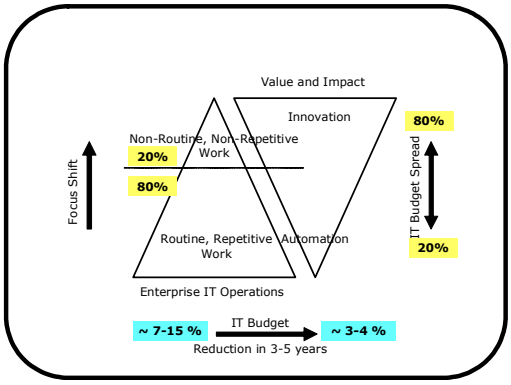

Fig. 4. Enterprise IT Renewal Indicators

We aim to automate as much as possible of the routine works freeing resources for more difficult and inventive duties. It should be mentioned, however, that 'self-managing enterprise IT' is more an ideal picture than technological reality. But it can be achieved only if time and resources are given to the IT stuff involved in hands-on operations. Certain class of difficult problems will remain forever in domain of unique human expertise. The mentioned principles are used for construction of multiyear renewal program guided mainly by the cost reduction and followed by business growth.

We believe that the research in Artificial Intelligence (AI) has addressed several problems relevant to 'self-managing' concepts and as such can serve as source of experience and inspirations for new class of self* technologies. Yet another research, in the Science of Complex Systems, has investigated models of very large scale, interacting systems and has dealt with complexity phenomena. As such, they are complementary domains by its nature and excellent starting point for the cross-disciplinary research.

Self-Management can be decomposed further into: self-configuration, self- re-configuration, self-healing, self-recovery, self-optimization, self-tuning, self-defense/patching etc. Each term cover so wide area that here is mentioned only for reference but it deserves careful study and exploration [5]. Overall effect self-management is seen as IT system adapting and always improving [6].

The aforementioned principles have been practiced within Hewlett-Packard and have given visible, proven benefits. Therefore, we believe that the reader will find an appropriate practical use of here exposed architecting principles despite the fact that this is an wide-brash, high-level view of the enterprise self-managing IT system described in a very short-paper.

Several approaches for self-managing systems have been proposed for operating systems (e.g. [7]), storage systems (e.g. [8]), and web servers (e.g. [9]). Broader approaches to various aspects of self-managing systems have been studied and are offered by some of the larger IT companies (e.g. [10], [11], [12], [13] [14]). Here we distil the concepts learned by these practitioners of self-management into four broad, over-arching principles that help guide the development of these systems.

We are also aware of the fact that the majority of the current efforts are focused on 'automation' aspect and technologies for self-management, while we pledge for the wider approach and different angle. By exposing architecting principles, we aim at bigger benefits and larger impacts.

Acknowledgements

We thank anonym reviewers for providing useful comments.

10. References

[1] J. Giarratano and G. Riley, "Expert Systems: Principles and Programming", PWS-Kent, Boston.

[2] J. R. Quinlan. "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, California.

[3] J. A Freeman, D. M. Skapura, "Neural Networks: Algorithms, Applications and programming Techniques", Addison-Wesley.

[4] K.A. Delic, "On Enterprise Architecture and Strategic Choices", http://www.datawarehouse.com/article/?articleid=3201, October 17, 2003.

[5] IEEE Internet Computing, "Autonomic Computing", vol. 11. pp. 18-48, January/February2007

[6] K.A. Delic and U.Dayal, " Adaptation in Large-Scale Enterprise Systems: Research and Challenges", ACM Ubiquity, vol. 5 , issue 23, August 2004. http://www.acm.org/ubiquity/views/v5i23_delic.html

[7] M. Seltzer and C. Small. Self-monitoring and Self-adapting Systems. In Proceedings of the Workshop on Hot Topics in Operating Systems, May 1997.

[8] Bandwidth Allocation in a Self-managing Multimedia File Server. In Proceedings of the Ninth ACM Conference on Multimedia, October 2001.

[9] P. Pradhan, R. Tewari, S. Sahu, C. Chandra, P. Shenoy. An observation-based approach towards self-managing web servers. In Proceedings of the Tenth International Workshop on Quality of Service (IWQoS 2002), May 2002.

[10] HP – OpenView Self Healing Services: http://h20229.www2.hp.com/news/about/index.html

[11] SUN – Predictive Self Healing: http://www.sun.com/bigadmin/content/selfheal/

[12] IBM – Autonomic Computing/Tivoli: http://www.research.ibm.com/autonomic/,http://www03.ibm.com/autonomic/

[13] Microsoft – Dynamic Systems Initiative: http://www.microsoft.com/australia/dynamics/crm/default.mspx

[14] Intl. Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007) http://www.seams2007.cs.uvic.ca/