SPIRIT: Service for Providing Infrastructure Recommendations for IT

Ashwin Lall
University of Rochester

Anca Sailer
IBM Research

Mark Brodie
IBM Research

Abstract

*We present SPIRIT, a **S**ervice for **P**roviding **I**nfrastructure **R**ecommendations for **I**nformation **T**echnology. SPIRIT allows maintenance support providers for Small-to-Medium Businesses (SMBs) to recommend solutions which are standardized (SMBs usually cannot afford customized IT solutions), flexible (accommodating as much as possible the customer's existing IT environment), and cost-effective (minimizing the cost of upgrading the customer's environment). SPIRIT works by first aligning the customer's IT infrastructure with a "template" describing the best practices recommended by the maintenance support provider. Then, the aligned environment can be upgraded by choosing from a standard set of well understood, highly automated (and therefore economical) options. In this paper we present the framework of our solution.*

1 Introduction

A Small-to-Medium Business (SMB), defined as having less than 1000 employees, usually cannot afford the elevated cost of highly customized IT applications and infrastructure. If an SMB customer needs to modify their IT infrastructure in order to correct a problem, or to maintain or improve over-all service availability and quality, their IT maintenance service provider needs to recommend solutions that (1) minimize the risk of unavailability by minimizing the number of changes needed in the customer's environment and by increasing automation, (2) maximize the benefit of applying the resolution, and (3) can be shown to be cost-effective for both the customer and the maintenance service provider.

In the past, IT maintenance service providers' efforts were mostly related to fixing their customers' hardware and software problems in isolation. Modern enterprise environments increasingly demand more sophisticated support that considers the whole IT infrastructure and its interdependencies. For instance, in a multi-tier e-commerce system, upgrading the application server may benefit the application business logic and fix its issues at the risk of introducing end-to-end performance degradation due to database overload or incompatibility. Existing solutions that enable maintenance support to provide more elaborate resolutions to the customer are primarily directed at specific resolution niches. For instance, performance problem resolutions focus mainly on run time provisioning [5], capacity planning [9, 10], or limiting traffic access [7, 8] in order to satisfy the service level agreements for the incoming traffic. Solutions related to cost-effective resolutions focus on optimization through server and storage consolidation [6, 12]. There are many approaches to problem resolution and cost optimization, however these do not provide a holistic maintenance service approach involving multiple resolution types for a given problem, nor do they optimize across different resolutions. Maintenance deals with a wide range of problem resolutions and what is needed is a way of comparing different possible resolutions in terms of their cost, benefit and risk, for both the customer and the maintenance support provider.

This paper seeks to address these issues. Specifically, methods are provided which examine failure notifications (both proactive and reactive), resolution rules, dependency constraints between IT subsystems, IT product costs, and best practices IT infrastructure templates to (1) optimize for the customer the cost-benefit ratio of the resolutions suggested by the maintenance provider and (2) minimize the service cost for the maintenance provider.

2 Providing cost-benefit recommendations

SPIRIT, our methodology for providing the customer with optimal recommendations, includes two *off-line* procedures and two *on-line* procedures. The off-line procedures collect and pre-process data that is then used in the on-line phase.

2.1 Off-line procedures and golden templates

The two off-line procedures are *collect* and *pre-process* (see Figure 1). The data gathered during the *collect* phase include (i) remedies recommended to solve known prob-

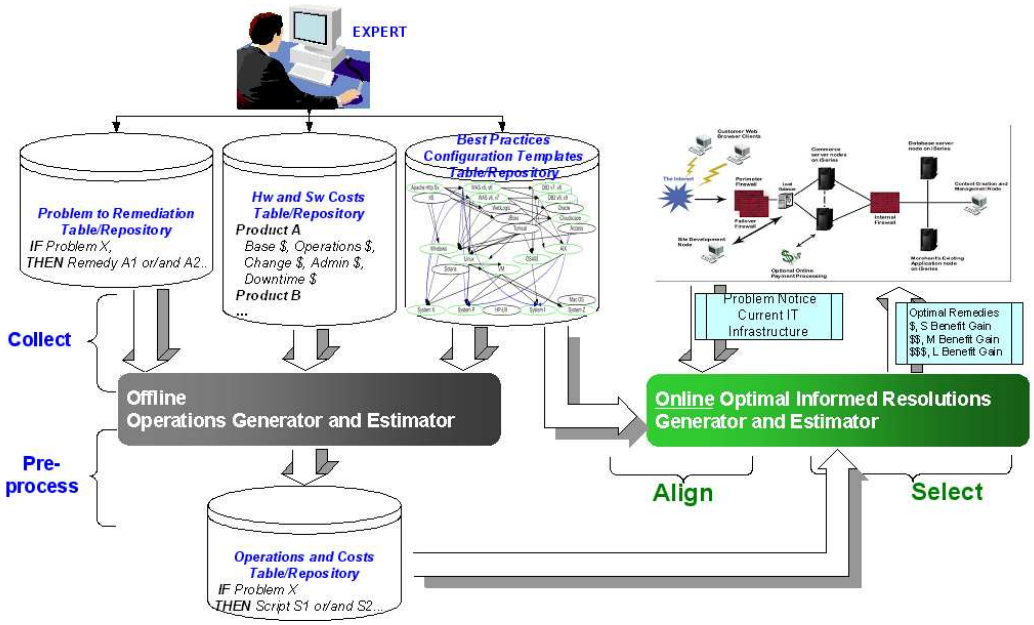

Figure 1. SPIRIT includes two off-line procedures and two on-line procedures.

lems — these can be obtained from manuals [3, 4], web sites [1], or forums [11], (ii) costs of the products supported by the maintenance service provider (these costs, which include direct costs like product cost, operations cost, administration cost and change cost, and indirect costs like downtime cost, are typically available for TCO calculation [15]), and (iii) IT constraints and dependencies known to exist between products supported by the maintenance service provider.

The *collect* phase is also used to build an IT configuration template describing a specific aggregation of best practices IT configurations recommended by the maintenance service provider, which we refer to as a "Golden Template" (GT). A maintenance provider may use different GTs, individualized per industry or per customer type. Figure 2 illustrates a simplified GT.

A GT includes IT dependencies and constraints that reflect the best practices configuration templates supported by the maintenance provider. The bold nodes in Figure 2 identify the products supported by the maintenance provider considered in our experiments. Examples of GT products and their dependencies are: "Web application server M version a.b.c works with database server N version x.y.z"; "Web application server M version a.b.c works on Linux Suze version n.m." The solid arrows in Figure 2 indicate such configuration dependencies. Lack of an arrow indicates either an unfeasible configuration or a constraint on a potential dependency that is unsupported. Note that not all valid dependencies and constraints are shown for visibility reasons.

A special type of configuration constraints are the classes of equivalence rules that indicate which products provide similar functionality. Examples of classes of equivalence rules are: "Web application servers are: WebLogic, JBoss, JRun, Tomcat," "Database servers are: DB2, Access, dBase, MySQL, Oracle, SyBase." In Figure 2, the nodes clustered together belong to the same class of equivalence.

The second off-line procedure is *pre-process*. Here the operations for performing common tasks such as software products installation, configuration, upgrade, migration and troubleshooting, particularly for simple problem resolutions, are standardized and automated — this is the advantage of having a GT instead of customized IT solutions, since the automated operations can be made cost-effective and their benefit evaluated. It is only because the GT reduces the number of possibilities so drastically that it is even feasible for us to consider automation of all these tasks. Doing this in the *pre-process* off-line procedure, prior to run time, reduces the run time labor cost of the maintenance service and the time to repair the problem.

2.2 On-line procedures and cost-benefit optimization

The core of SPIRIT consists of the two on-line steps, *align* and *select* — see Figure 1. The on-line procedures take place after a problem reported by a customer (or, proactively, a potential failure) has been identified and the root cause determined. In the first step SPIRIT *aligns* the customer's current IT infrastructure with the maintenance service's supported products by making the customer's infrastructure consistent with the provider's GT, using the minimum number of changes and taking into account any special

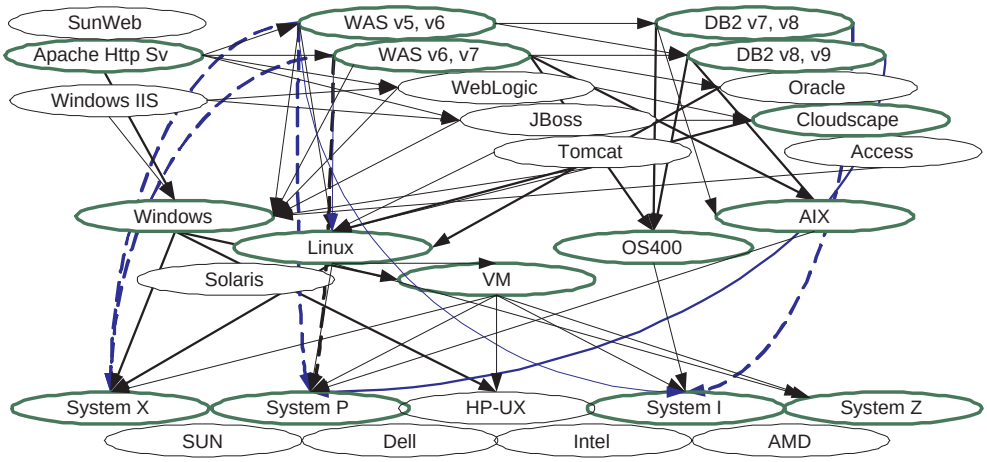

Figure 2. Example of Golden Template as a specific instantiation of best practice IT infrastructure.

customer restrictions. This optimized migration is the major challenge for the maintenance service provider because of the multiple potential customer restrictions, e.g., minimum costs of changes, minimum or no changes of the products directly related to their applications, software restrictions, etc. Moreover, this optimization is the key enabler of the optimized remedy recommendations service since it makes possible the use of the automated remediation operations built off-line for the GT.

Note that most IT infrastructures rely on redundancy at any of the stack layers, e.g., application, middleware, hardware. In such cases, we consider for the migration optimization only the base pattern of the IT infrastructure, with unique products, rather than the whole IT environment with duplicates. The results of the alignment are applied seamlessly to all the infrastructure products, duplicates or not.

The second on-line step consists of *selecting* from the multiple identified resolutions for all target infrastructures the optimal ones, from a cost-benefit tradeoff perspective. The advantages of our solution for the customer are that for each resolution option the recommended resolution is optimized with respect to the cost and benefit of the resolution. The advantage of our solution for the maintenance support service lies in reducing the maintenance cost by first aligning the customer's IT infrastructure to one of a limited number of best practices templates and then applying the chosen resolution using a standardized, highly-automated, and hence inexpensive process.

3 Formulation

In this section we formally introduce the notation and formulate the problem of aligning the customer's IT environment with a GT.

3.1 Notation

We represent our universe of objects as $V = \bigcup_i C_i$, where the C_i are pair-wise disjoint classes. Each class represents a set of objects that can be substituted for one another and designates a class of equivalence as described in Section 2.1.

We are given the GT graph $G = (V, E)$, which expresses the relationships between objects in V. An example of such a graph is given in Figure 2.

We are also given the customer environment as a graph $S = (V', E')$, where V' is the set V with some objects appearing zero or more times. That is, for any node $v \in V$, V' may have k copies of this node represented as $v^1, \ldots, v^k$, for some k. We will refer to these as separate *instances* of the same object. Since a particular product may appear several times in the customer environment (e.g., there may be multiple copies of WAS v6 running on different machines), we use this concept of instances to distinguish them. The dependencies in the customer environment can be obtained using an automatic dependency discovery tool [2].

Objects within a class can be substituted for one another, and for this we define a *substitution cost function* called $c : V \times V \rightarrow \mathbb{R}$, where $c(v_1, v_2) = \infty$ for v_1 and v_2 in separate classes. The substitution cost is computed by taking into account all relevant costs, such as product costs, administration costs, downtime costs, and weighting them appropriately to reflect the customer's preferences or criteria of optimization. We extend the cost function to the domain $V' \times V'$, treating each instance identically to the original.

Lastly, we have an *edge cost function* $e : V' \times V' \times V \times V \rightarrow \mathbb{R}$ which represents the cost of replacing a link between two objects in the customer's current environment by a link between two objects in the GT.

Note that the substitution and edge costs are among the

data collected in the off-line phase of SPIRIT described above.

3.2 Problem Statement

Our goal is to transform the current customer environment graph S to a target graph T by replacing some vertices in S by objects in the corresponding class. The constraint that we place upon T is that every edge in it must also be an edge in the GT graph G. Our goal will be to find the minimum cost transformation (assuming that one exists).

More formally, we want to find a function $f : V' \rightarrow V$ that re-labels the vertices of S in such a way that, for every edge $(v_1, v_2) \in E'$, we have that $(f(v_1), f(v_2)) \in E$, and such that the total cost

$$\sum_{v \in V'} c(v, f(v)) + \sum_{(u,v) \in E'} e(u, v, f(u), f(v))$$

is minimized.

Due to lack of space, we do not present the full solution and evaluation here. For all the details, please refer to our technical report version [13].

4 Related Work

As discussed earlier, most existing solutions have narrow applicability, such as run time provisioning [5], capacity planning [9, 10], restricting traffic access to maintain service level agreements [7, 8], and server and storage consolidation [6, 12]. SPIRIT, on the other hand, has broad applicability to the entire customer environment. Additionally, SPIRIT provides multiple resolution options, each with cost/benefit tradeoffs.

In [14], a similar problem is addressed in the context of migration in Service Hosting Environments. The solution proposed is to use a model, called the System Service Configuration Model, to describe the dependencies and configuration parameters. SPIRIT avoids dependence on a model (which may need to be updated frequently) by making use of automatic tools such as the Tivoli Application Dependency Discovery Manager (TADDM) [2] to obtain its data.

5 Conclusion and future work

We presented a maintenance service method for offering optimal resolution options for issues in the customer's IT environment. In order to provide the most suitable solution for the customer's business, information such as, but not limited to, the cost, benefit and complexity of change is evaluated by SPIRIT for each resolution option and the optimal cost-benefit resolutions are provided to the customer for selection. Additionally, we limit the range of potential resolutions by aligning the customer's IT infrastructure to a limited number of best practices templates, thus enhancing the reliability of the customer's environment and reducing the maintenance cost.

The advantages of SPIRIT for the customer are that for each resolution option the recommended resolution is optimized with respect to the cost and benefit of the resolution. The advantage of SPIRIT for the maintenance support service lies in reducing the maintenance cost by first aligning the customer's IT infrastructure to one of a limited number of best practices templates and then applying the chosen resolution using a standardized, highly-automated, and inexpensive process.

Acknowledgments: The authors would like to thank both Krishna Garimella, for providing them with data, and George Galambos, for his invaluable guidance.

References

[1] IBM Support. http://www.ibm.com/support/troubleshooting/us/en/.
[2] TADDM. http://www.ibm.com/software/tivoli/products/taddm/.
[3] DB2 Warehouse Management: High Availability and Problem Determination Guide. SG24-6544-00, Redbook, 2002.
[4] Websphere application server v6 problem determination for distributed platforms. SG24-6798-00, Redbook, 2005.
[5] C. Adam, R. Stadler, C. Tang, M. Steinder, and M. Spreitzer. A service middleware that scales in system size and applications. In *Integrated Network Management*, 2007.
[6] M. Badaloo. An examination of server consolidation: trends that can drive efficiencies and help businesses gain a competitive edge. *White paper on IBM Global Services*, 2006.
[7] B. Callaway and A. Rodriguez. Enable XML Awareness in WebSphere Extended Deployment With WebSphere DataPower SOA Appliances. *White Paper on IBM developerWorks*, 2006.
[8] Y. Diao, J. Hellerstein, and S. Parekh. Stochastic modeling of Lotus Notes with a queueing model. In *Computer Measurement Group International Conference*, 2001.
[9] M. Goldszmidt, D. Palma, and B. Sabata. On the quantification of e-electronic commerce. In *EC*, 2001.
[10] E. Hubbert and J.-P. Garbani. *Sustaining Application Performances: The Capacity Planning Software Market.* Forrester Research, 2007.
[11] IBM Website. Forums and community. http://www.ibm.com/developerworks/.
[12] A. Kochut, K. Beaty, and N. Bobroff. Dynamic placement of virtual machines for managing SLA violations. In *Integrated Network Management*, 2007.
[13] A. Lall, A. Sailer, and M. Brodie. Spirit: Service for providing infrastructure recommendations for it. Technical Report RC 24466, IBM Research, 2007.
[14] Q. Ma, Y. Li, K. Sun, and L. Liu. Model-based dependency management for migrating service hosting environment. In *IEEE International Conference on Services Computing*, 2007.
[15] T. Pissello. *The Business Value of HP-UX 11i: HP-UX 11i on Integrity Servers vs. IBM AIX 5L on eServer*. Alinean Inc., 2006.