ERMIS: Designing, developing, and delivering a remote managed infrastructure services solution

M. Viswanathan
H. Shaikh
A. Sailer
Y. Song
X. Fang
Y. H. Wu
Z. L. Zou
K. P. Reddy
A. Deshmukh
M. Gupta
B. Krishnamurthy
M. Sethi
B. Viswanathan
J. G. Gulla
F. Matar

Remote management of IT (information technology) infrastructures, as a service, has received significant attention due to the numerous benefits it offers. In this paper, we discuss our work on taking a previously India-only remote IT infrastructure management service offering to a global audience. We first provide an overview of this IBM offering, called Express™ Remote Managed Infrastructure Services (ERMIS), and then focus on two specific aspects, service catalog and problem determination and resolution (PDR). In service catalog-based remote management, customers browse the catalog and place orders as they would using an online store. Our service catalog entries contain workflows to automate the deployment of requested applications or the configuration change of customer resources. High-level customer requirements are translated into capacity sizing and configuration parameters, which are then used to provision the final solution by making use of ITIL® (Information Technology Infrastructure Library) release and configuration management processes. PDR involves detecting, locating, and fixing anomalies. Probe-based techniques are used for targeting the problem location, while knowledge management assists with problem analysis, diagnosis, and classification. ERMIS architectural decisions have been driven by emerging-business requirements related to small and medium-sized businesses.

Introduction

The market for remote infrastructure management (RIM) is a growing rapidly. Typical RIM service providers use their own fixed network operations center (NOC) to remotely monitor and manage clients' IT infrastructure elements such as networks, system hardware and software, operating systems (OSs), and various applications. Performance is measured against customer-defined service-level objectives. The primary business value for RIM services arises from the fact that it helps global enterprises as well as small and medium-sized businesses reduce costs for infrastructure management and, at the same time, gain access to expert skills. Several large and small IT vendors, both national and multinational, focus on such services. The principal differentiators among vendors include the business models they use, the underlying tools and technologies, and the skills and processes used in delivery.

IT suppliers are aware that the RIM services industry is growing with a rich potential for innovation and profit through the use of automated tools and a global labor force. Traditional management approaches use technical personnel dedicated to a specific customer, whereas RIM makes use of a shared labor pool. This idea of using one person or a group of skilled individuals serving a large number of customers makes it possible to employ a diverse set of skills and utilize them more efficiently than using previous approaches. Many companies have embraced the RIM approach to provide IT services to customers. Wipro delivers its infrastructure management services through a Global Command Centre (GCC) present in India with a point of presence located globally

0018-8646/09/$5.00 © 2009 IBM

that performs basic event management and is integrated into a GCC. Patni Computer Systems Limited [1] has global NOCs supporting multiple monitoring consoles that proactively manage remote-client environments and more than 10,000 applications, databases, or servers, as well as 20,000 Simple Network Management Protocol (SNMP)-based devices [2–5].

RIM is suitable for establishments that have an investment in networks, servers, middleware, and applications that are already located on their premises or at a third-party data center, and that require monitoring, managing, and reporting of their infrastructure in a standardized, scalable, and cost-efficient manner, with minimal transformation and no transfer of people or assets. The term "monitoring" includes availability and performance monitoring, reporting, problem analysis, and notification. There is no remediation but only an indication of the issue noticed, with the onus on the end-user to mitigate either the issue or its root cause. "Management" is an all-encompassing term covering all ITIL** (Information Technology Infrastructure Library) function terms that frequently suffixed with the word. The management component is often structured in different ways, ranging from fully remote operation to remote operation with an on-site component. For the purposes of this discussion, the wholly remote model is the only approach that will be discussed hereafter.

Key RIM functions address IT failure. When IT systems fail, the problems do not always manifest themselves as a total system failure. Instead, the end-user experiences, in many cases, a degraded performance. For example, a condition such as "CPU workload is high," "CPU is offline," or "processes consuming high % of CPU use" may be indicative of a deeper problem that should be investigated by an administrator with the assistance of an application specialist. In other cases, a problem may be a transient issue caused by heightened application activity. RIM monitoring of IT assets can be performed either by availability monitoring or performance monitoring. Availability monitoring provides a binary result—either the device is up (operational) or down (nonoperational). Usually, this class of monitoring is performed every three to five minutes. Performance monitoring involves extracting a set of key performance indicators (KPIs) and then measuring these against a preset threshold. The remediation or the decision to intervene may depend on the type of IT asset and how critical it is to the end-user's business.

Once the issue has gone beyond the warning stage (as determined with respect to a threshold) or when the underlying IT system has stopped functioning, a reactive element of systems management assumes control. RIM management functionality is a complement to monitoring and implies managing and repairing of the IT system, fixing the underlying issue that caused the failure, performing root-cause analysis to determine the source of the failure, and applying remedies that avoid future failures. **Table 1** summarizes the pairing of monitoring and management with their proactive and reactive dimensions. Agile tools can have a very small effect on system performance and yet serve to proactively alert the concerned delivery engineer about impending problems, as discussed later in this paper.

The next section delves broadly into one RIM example, namely ERMIS. We describe the individual components of ERMIS as they are implemented today, beginning with the overall operational framework. Next, the ITIL management tasks supported by ERMIS are presented with their required skilled resources, followed by the automation that is planned to be introduced in phases. Viewing the current state of the industry, one notices that the areas with the least advances in the filed involve the service request catalog and problem determination (PD).

The services request catalog involves an area of heightened interest to IT vendors. This catalog is a list of items that may be obtained by an IT customer. This is presented as an ordered family of lists and achieves two goals. First, it simplifies the IT shopping experience by offering a clearly articulated list of most commonly requested services. Second, it allows the service provider to make small adjustments to its offerings involving a closed set of IT services—services that can be packaged as repeatable units of algorithms combined with a process. Automation of management tasks for such commonly requested services will help in lowering costs while quickly delivering leading-edge services to customers.

Problem determination and resolution (PDR) represents a substantial part of operational costs; therefore, faster and more effective PDR contributes to a substantial reduction in system administration costs. While ERMIS incorporates full automation of monitoring—involving event detection, breach measurement, assignment of severity and impact levels, ticket generation, portal update, and customer notification by e-mail—automation of management functions remains elusive. Given that there are management tools that partially automate functions such as software distribution or service restart, automation of broad IT areas such as PD is still rare. This is currently a manual task that requires an individual's analytical skill and experience. However, in a service that operates continually throughout the day and week, operational consistency is essential, and diagnosis of critical business systems should not be the responsibility of a single individual.

We have chosen to pursue these two management automation areas for our near-term research. We report

Table 1 Monitoring and management.

Dimension	*Monitoring*	*Management*
Proactive	Detection—CPU workload is high or CPU is offline. Processes consuming high percentage of CPU use.	Prevention—Paging space administration. File space content and container management. Process recycling—Problems can be mitigated before they result in a failure.
Reactive	Action—Events or messages result in activity taking place (e.g., restarting failed processes or sending emails or messages to key personnel).	Action—Analyze symptoms on a device that indicates high resource consumption rates or atypical behaviors.

on our approach and progress in the sections "Service automation using service catalog" and "Problem determination and resolution."

ERMIS

This paper uses IBM Express* Remote Managed Infrastructure Services, or ERMIS, as an example for all remote managed services. **Figure 1** shows the IT system view of the ERMIS architecture. The service includes a secure network connection between the ERMIS Global Delivery Center (GDC) and the customer premises. ERMIS is a remotely delivered service, so the majority of the infrastructure hardware and software is hosted at the delivery center. The sole customer-site infrastructure is an optional data collector that polls the IT endpoints on the customer's local area network. Monitoring and management are performed over the private or public wide area network (WAN). We may now examine the different components of this architecture.

Architectural components

Figure 1 shows the various architectural components that constitute the ERMIS delivery framework. The ERMIS GDC comprises a set of shared monitoring and management tools that provide the base monitoring, reporting, and notification services. This is supported by a staff of engineers who provide level 1, level 2, and level 3 support representing multiple levels of expertise and are the unifying factor between all the services provided from this center. This zone represents the ERMIS intranet zone. Access to the world outside of this zone is via a WAN, which can be private (e.g., MPLS, or multiprotocol label switching) or public (e.g., the Internet). This shared connectivity is used for both monitoring and management services. Isolation is provided by creating a distinct and secure virtual private network (VPN) from the delivery center to each customer's edge device. This same channel is used to perform SNMP polling, to provide system access via telnet, or to run remote desktop protocol. The customer premises shown represent a data center or branch office location where customer endpoints are located. A customer may have a single location or multiple sites, and in the latter case, each site must be reachable from the delivery center.

The ERMIS NOC-Inside (NOC-I) Monitoring toolset is a fully automated component that provides event monitoring, threshold configuration, and breach monitoring (via SNAPPiMON [6]), reporting (via SNAPPiMON), incident ticketing (via coDesk [7]), asset management (via Asset Manager [8]), reporting (via NOC-I Customer Portal), and notifications (part of coDesk). These are supported by engineers who provide incident, problem, configuration, and changes according to ITIL guidelines and IBM best practices.

ITIL management support in ERMIS makes use of tools that are native to the systems and devices and well-documented processes applied by the support staff. The support staff members have varying levels of expertise, from the most general (level 1) to the more specialized (level 3). Level 1 staff members are first-level personnel reacting to auto-generated incidents and service requests. These may be simple requests pertaining to the portal (e.g., a request for a new user ID) or KPIs (e.g., add or remove performance indicators) and attendant threshold levels (raise or lower levels). Level 1 staff performs simple tasks within a spectrum of management tasks including reacting to auto-generated incidents, problem analysis, and recommendations regarding persistent IT incidents.

Level 2 support personnel represent the first level of personnel reacting to persistent problem and change management queries and issues. These incidents escalate to become problems in the ITIL sense. This requires some expertise in an area of specialization—expertise such as applying preexisting scripts and performing problem analysis and some of the more obvious root-cause analyses. They also represent the second-level personnel reacting to auto-generated or explicitly requested severity levels 2, 3, and 4 related queries. Level 2 personnel are responsible for augmenting and applying reusable solutions from the knowledge base, logging onto customer devices and systems, and fixing problems and restoring systems to working condition. However, sometimes even more specialized knowledge in the form

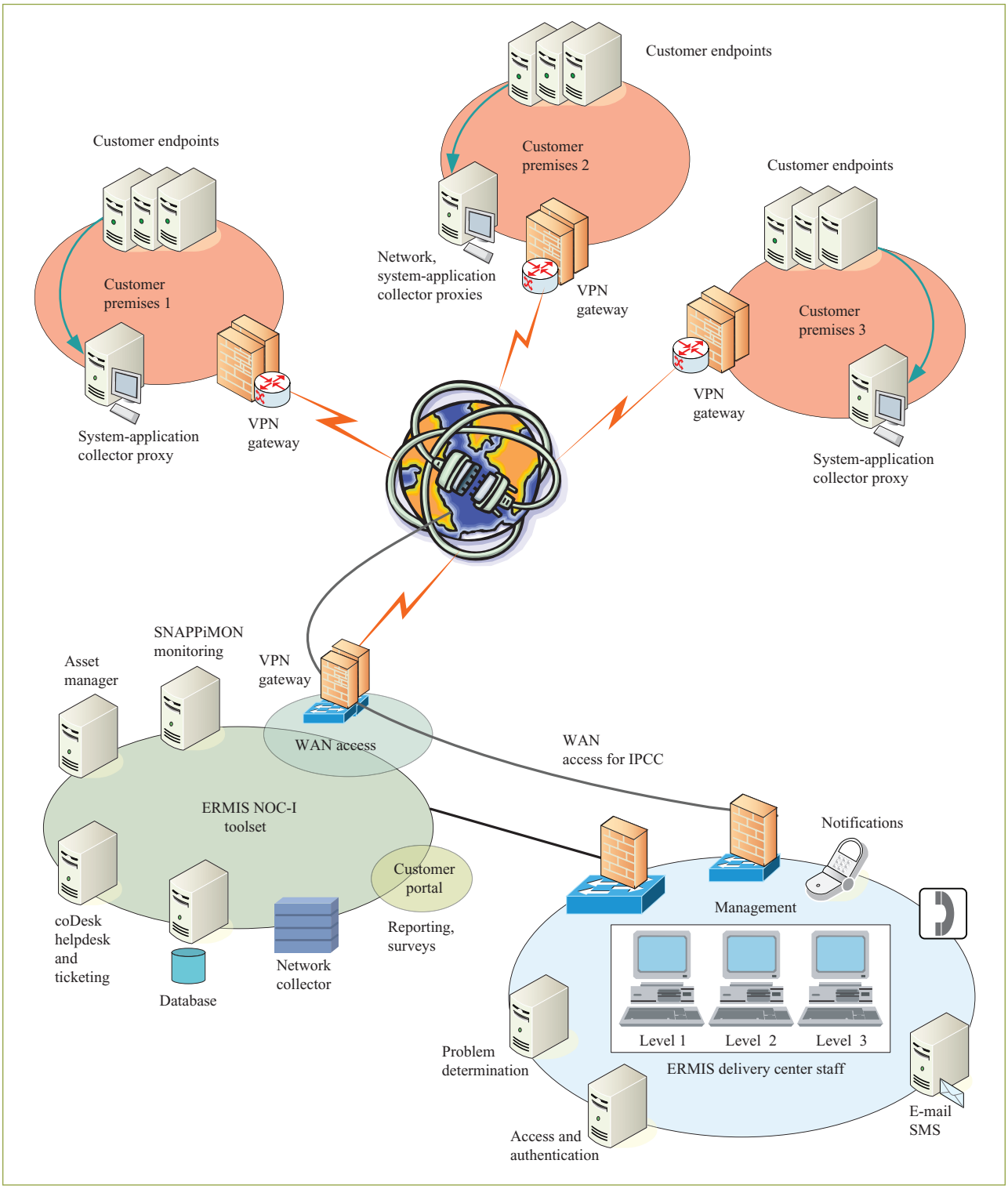

Figure 1

ERMIS architectural component diagram: The central symbol in this diagram represents wide area networks (WANs), for which the most common example is the Internet. (VPN: virtual private network; IPCC: incident, problem, change, and configuration management; SMS: short message service; coDesk: ERMIS help desk application; NOC-I: Network Operating Center-Inside; SNAPPiMON: ERMIS monitoring application.)

of level 3 support engineers is required. These engineers provide supervisory support, create the scripts that will be applied by others, and organize and maintain the knowledge base that stores common methods and procedures. Level 3 staff may also perform second-level staff roles, reacting to problem and change management queries and issues.

The ERMIS Customer Portal is a web-based portal that provides customers and end-users a dashboard view of their IT environment, provides access to scores of availability and performance reports, serves as a single source for creating and viewing the status of open tickets, and manages IT assets. The customer portal is configurable to create graduated access to four inherent applications based on the role an individual plays within a customer organization. It is also configurable to create a cumulative view that logically aggregates disparate IT environments into a single logical set based on geography or some other consideration.

Monitoring

ERMIS monitoring follows a distributed model consisting of a centrally located manager supervisor, supported by distributed network and system-application collectors. Monitoring consists of two parts. The first part involves availability monitoring, which provides information on whether the end device is functioning, thus returning a binary response. The second part, performance monitoring, is more nuanced and goes beyond binary values to provide information that ranges from CPU (central processing unit) utilization to response times. Additionally, many dozen performance indicators are captured during a performance monitoring. Availability measurements are made frequently, generally every three to five minutes, whereas performance polling is done less frequently. An unavailable device will get immediate attention, whereas a heightened performance indicator may just elicit a closer watch for the next data-gathering cycle or two. The network collector is responsible for polling all customer endpoints for up or down status (i.e., availability) and polling all customer network endpoints for performance. This differential monitoring involves a simple division of labor. For the same number of devices, network collectors have a smaller workload, and hence, the task of measuring availability is performed by these collectors.

A network collector uses the Internet Control Message Protocol (ICMP) for availability monitoring. This ICMP message is more commonly known by most computer users as a *ping*. Within each availability window (every three to five minutes), one or more pings are sent to each customer device. Any one response from each endpoint is considered a signal that it is active. This is a "loose" algorithm because if the endpoint in question returns only one of N pings (if the tool is programmed to send N pings per measurement cycle), then the device is considered to be "available." "Tighter" (e.g., more stringent) rules may be applied for more business-critical endpoints; for instance, the rule might be that at least $N/2$ responses be returned for every N pings of any one destination device. Performance monitoring of devices is done by polling IT endpoints for all the defined KPIs every 15 or 30 minutes. The protocol used is Simple Network Management Protocol (SNMP) [9]. Devices that support SNMP are network devices (e.g., routers, switches, firewalls, lease lines, unified communication devices, and IP devices), printers, uninterrupted power supply systems, and some storage devices. The tool supports SNMP versions 1, 2, or 3. It also supports SNMP 1, 2, and 3 listeners if the endpoint is capable of deploying SNMP agents that proactively detect and set "traps." While SNMP polling is initiated by the collector, traps are initiated by the agents on the endpoints.

A network endpoint that is required for ERMIS to function is a VPN-capable edge device at the customer's location. A secure "tunnel" is set up from the firewall of the ERMIS delivery center to each customer's VPN device. This is a configuration task that requires matching parameters on both devices and coordinated effort at both ends to allow the devices at each end to reach the other using the internal IP (Internet Protocol) addresses. Essentially, the configuration allows the monitoring server at the delivery center to reach and address every device at the customer location. To explain why this is necessary, we must consider the collector architecture. SNAPPiMON collectors, by constructions, are multi-tenant and can address, support, and segregate multiple customer networks. By definition, therefore, the collectors can be placed at the delivery center to poll one, many, or all customer environments. The network collector, which has twin duties as outlined above, can be placed at the delivery center. Further, since multiple network collectors can coexist, the optimal collector placement involves one network collector at the delivery center and one at the customer site. In order to understand what may first appear to be task duplication, we discuss the role of the system-application collector.

The system-application collector gathers performance data from servers, OSs, middleware, databases, and applications. It uses the native protocol of the endpoints to retrieve performance information from the registry or equivalent information stores. Protocols include from Windows Management Instrumentation (WMI) and Performance Data Helper (PDH) for Windows, telnet and secure shell (SSH) for Linux**, and Java** Database Connectivity (JDBC) for databases. Of these, WMI and PDH are very verbose, with valuable performance data representing about 1% of the total data exchanged

between collector and endpoint. When many such devices are concentrated at a location or customer site (and this kind of concentration predominates in the small and medium business market), this concentration tends to increase bandwidth requirements and extend polling windows. Hence, the recommended solution is to place the system-application collector on a workstation at the customer site within the customer's LAN. The collector, therefore, does the communication-intensive polling transactions locally, removes the extraneous data, applies the thresholds (which are always stored on and applied from the collectors), and then sends the requisite information through the secure channel back to the delivery center manager, which in turn processes the events and transmits the remainder to the data store and the portal.

On the other hand, the network collector uses SNMP, which is a fairly concise protocol, and can be operated centrally from the delivery center or locally from the customer site. For consistency, ease of maintenance, and efficiency, the network collector is best co-located with the system-application collector workstation. In this configuration, only the collector workstation is required to be accessible within the VPN tunnel. Therefore, only the IP address of the collector workstation needs to be network address translated (NAT) to a unique address assigned by the delivery center. (Without the network address translation, the collector will be assigned IP addresses by the customer's IT team, and there is a risk that two unrelated collector workstations at two customer sites may be assigned the same IP address. This will cause confusion at the delivery center when the monitoring tool accesses a collector to gather polled data.) In this model, both the network and the system-application collector perform in exactly the same way vis-à-vis the customer IT endpoints from the same physical location.

One may ask why we retain a delivery center-based network collector? Network devices are often the entry point into systems management for many small customers. This means that ERMIS faces many usage scenarios that consist of entirely SNMP-based devices that are to be monitored or managed. In these scenarios, the justification for procuring a collector workstation becomes tenuous. If we also consider ease and brevity of ICMP-SNMP monitoring, then the option of monitoring from a remote location becomes very attractive to the customers. However, there are several drawbacks. All polling is done remotely, and any break in the link will place the customer site in a state without any monitoring. Additionally, since availability monitoring is now done remotely, all of the customer endpoints must be NAT. This translation is not terribly onerous for OSs that are script or command line based.

The overall monitoring architecture may be considered as a network collector for customers who have no system- or application-related IT, a local system-application collector (at least one per customer for up to 400 endpoints) for all customers who do have systems and applications to be monitored, and a local network collector (at least 1 per 1,000 endpoints) co-located with the system-application collector for those customers not being served by the central network collector.

Event management

Event management is the ongoing activity to collect, monitor, and analyze event notifications and to take appropriate actions to respond to those events. An event is translated to a status change of a resource. Event notifications are messages that are in various formats based on the event source. These formats include SNMP traps, log entries, or TCP (Transmission Control Protocol) communications using proprietary and standard protocols. Event correlation is performed to determine whether events are duplicated and whether they are expected or unexpected. Automating the event management process makes it efficient and reliable and drastically reduces the implementation cost for console operation services. The ERMIS event management system integrates events of network, system, database, middleware, and application management on the same infrastructure, as shown in **Figure 2**. The SNAPPiMON collectors act as central collection points for event notifications from a variety of sources. These are accumulated, digested, and presented via the SNAPPiMON manager tool.

Asset Manager and help desk

The Asset Manager application records, identifies, controls, and tracks all assets in customer environments. Asset Manager is an ITIL-compliant configuration management database (CMDB), is integrated with the ERMIS help desk system coDesk, and provides insight into issues, incidents, change requests, and problems associated with each asset. Therefore, a consolidated list of incidents, problems, and changes can be viewed across all assets and information including ticket details and audit history. IT and non-IT assets are managed across the complete life cycle including ordering, installing, using, repairing, upgrading, and discarding. A data structure to create and maintain the customer eligibility (e.g., for monitoring versus management, and for the number of changes allowed while considering what has been applied) is managed in this application on a per-asset basis.

The help desk component, coDesk, is a Web-based, ITIL-centric help desk for managing IT and non-IT issues. It provides automation and support for key ITIL-

recommended best practices for incident, problem, change, and configuration management processes. Automation for incident management includes flexible routing based on incident category, classification, priority, impact, data, and time. It can track service-level agreements (SLAs) for response and resolution times. Additionally, incidents can be related to problems, change requests, and configuration items. Problem management capabilities include support for root-cause analysis, workarounds, temporary fixes, and solutions. Change management is supported via Web-based logging of change requests by end-users and incident, problem, and change teams. Configuration management is supported via an extensible CMDB that can store standard and custom relations among configuration items, IT and non-IT assets, end-users, owners, maintainers, locations, departments, and projects. It integrates with Asset Manager to track predefined asset types such as desktops, servers, switches, routers, OSs, applications, and firewalls. Usage-limit notifications that arise from customer use of assets or changes to assets are triggered via the integration between Asset Manager and the help desk applications. coDesk also includes a full-featured knowledge base that allows self-help, self-service by end-users, and reuse and troubleshooting tips for the help desk staff.

SNAPPiMON and coDesk activities are also tightly integrated. An event detected by SNAPPiMON with respect to any given customer asset is passed to coDesk with SNAPPiMON-assigned severity levels, such as warning, critical, and fatal (for performance) and up or down (for availability). coDesk receives the message, looks up the asset name, finds the preconfigured severity level from Asset Manager, automatically assigns the severity level to the event, and generates a ticket. When coDesk generates notifications to customer-provided e-mail addresses (and SMS, or Short Message Service, destinations), the severity level is included in the mail. In the ERMIS service design, the severity level determines the notification type: Severity-1 events require notifications via phone, SMS, and e-mail; severity-2 events are notified via SMS and e-mail; and severity-3 and severity-4 event notifications are via e-mail alone. E-mail and SMS notifications are delivered via the Simple Mail Transfer Protocol (SMTP) gateway, which is integrated into the coDesk messaging template. Phone calls are reserved for catastrophic events such as the failure of a switch connected to 20 servers or for failures associated with the main customer link or router.

Customer portal

This Web-based portal is the single logical point of access for clients to ERMIS support tools and information. This component will be used to provide access to systems such as an executive dashboard, monitoring details, service desk functions, asset management, knowledge base, real-time and historical reporting, availability and performance statistics, user self-help, and user management. It features a real-time dashboard, a dashboard for historical reports, controls for a prescheduled e-mailing of reports, threshold and device alerts, notification with respect to availability events and performance threshold breaches, correlation between endpoints, viewing of escalations, and topological views of the customer environment, and other features. Along with an executive dashboard, it provides access to the monitoring, help desk, and Asset Manager applications in

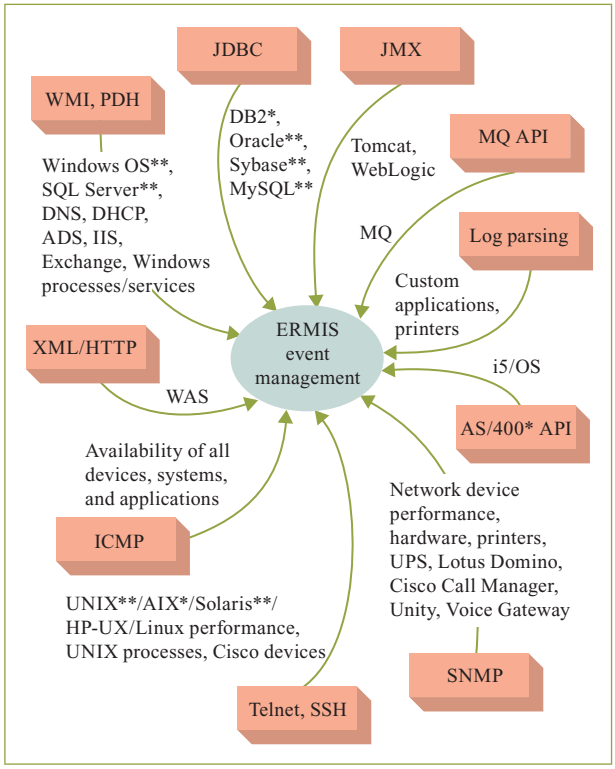

Figure 2

Event management: Apache Tomcat is a servlet container developed by Apache Software Foundation. Oracle WebLogic consists of a J2EE (Java Platform, Enterprise Edition) platform product family. (WMI: Windows Management Instrumentation; PDH: Performance Data Helper; SSH: secure shell; JDBC: Java Database Connectivity; ICMP: Internet Control Message Protocol; SNMP: Simple Network Management Protocol; JMX: Java Management Extension; OS: operating system; MQ API: message queue application programming interface; DNS: domain name server; DHCP: Dynamic Host Configuration Protocol; IIS: Internet Information Services; SQL: Structured Query Language; ADS: active directory server; WAS: IBM WebSphere Application Server; UPS: uninterruptible power supply.)

one portal. While the default is to provide user access to all the applications via a single user ID and password, it also has the capability to separate access privileges by application. The portal is the recommended forum for interaction with the customer. While e-mail and SMS notifications exist, they serve to merely draw the customer to the wealth of information that is available via the portal. In addition to obtaining performance and other analysis information relating to the health of a customer's IT, users can open tickets via the portal, exchange information including files and attachments via the ticketing portal, and maintain and store information relating to their assets.

Management task automation

The presentation of the service request catalog and PD are manual tasks in most manifestations of remote management services today. The service request catalog is essential to allow the customer to select a service from a fixed catalog of well-understood and well-defined services. This is the IT management equivalent of mapping a route between two locations on Google Maps** or selecting a book on a particular topic from the Amazon.com categorized catalog. This forces the service provider to advertise only those services that it offers and removes undue dependence on the skills of the delivery staff while making use of standard, leading-edge, and branded capabilities. In our approach to service catalog-based remote management, a customer-centric service catalog exports well-defined, simple, and composite services. Customers who subscribe to this portal browse the catalog and place orders in a manner similar to ordering at an online e-commerce site. An actionable service catalog entry contains prefabricated workflows to automate the deployment of requested applications or perform configuration changes on customer resources. A provisioning appliance is utilized at the customer site to act as an agent for server discovery and software installation. Our approach uses the concept of service classes, which provide discrete configuration points, embedded in the service catalog. Monitoring data from the NOC-I toolset is periodically used to facilitate learning and the provisioning of end-to-end solutions that more accurately match the user's requirements.

The second task for which automation is a challenge is PD. This is largely an intellectual exercise today that uses judgment and experience at the cost of repeatability and scalability. PDR is the process of detecting anomalies in a monitored system, locating the responsible causes, and fixing the problem. The main focus of our work is to effectively integrate and enhance the available NOC-I remote monitoring infrastructure with probe-generating techniques for better targeting of the problem location. Further, these probes are easily composable into new ones and usable by low-skilled personnel who manage the SMB (small and medium-sized business) IT environment. Knowledge management assists with problem analysis and diagnosis and guides the service request submission. We are building a framework that allows people to build PD plans consisting of probes and composite probes and then to deploy it in a customer environment (or even a data center) without requiring any change in the PD plans. In the following sections, we discuss the service catalog and PD.

Service automation using a service catalog

From a customer perspective, a service catalog is the most visible component of IT services. It is more than just a listing of services. A service catalog aids in providing functionality across different ITIL processes such as service catalog management; service-level management; and release, configuration, and change management. Services can have relationships with other services, and services may have a dependency or prerequisite relationship with other services. Further, a service can be composed of multiple other services and can also be bundled with other services. For instance, a core service and set of related services can be offered together, such as would be the case for an OS installation service and related system management services can form the OS service.

A service should be designed with a level of granularity that captures specific information but without needless complexity. The service definition typically includes capability, value of the service, pricing, and fulfillment steps as well SLAs to track the quality of delivery. This helps in standardizing delivery of services to multiple customers while allowing customization to suit individual customer needs. The various items that make up a service definition are document information, SLA for the service, and ordering and fulfillment information. Ordering can be with respect to procedures and pricing, a fulfillment into job plan, tasks, escalation, delivery SLA, workflow, images, and scripts.

Automating the delivery of IT services reduces the cost of delivery, time to delivery, and human errors, and helps in compliance checks and overall life-cycle management of services. Automation can occur at multiple levels. Process automation works at the execution layer and is concerned with deployment of service on an endpoint. Workflows that are part of the service definition capture the sequence of steps involved in automating these activities. A service catalog can be integrated with process management tools such as the IBM WebSphere* Process Server for business processes and IBM Tivoli* Process Manager for IT processes to execute and monitor workflows.

Service catalog in coDesk

As mentioned, coDesk is the service desk component of ERMIS. Both end-users and delivery staff access coDesk using a Web interface and are presented with role-specific views of activities that they can perform. Asset information is stored as configuration items in the asset repository and is linked to entities in the CMDB. Assets that are spread across different organizations and sites can be mapped hierarchically to location information.

The service catalog component in coDesk is integrated closely with the existing asset repository and CMDB to extend the capability of coDesk. The service catalog schema for IT services is concerned with software packages, and OS and configuration management activities. Software services are classified as *software package* for individual software applications, *operating system* for OS images, *software stack* for a collection of software applications, *distributed applications* for software solutions that can be deployed across multiple endpoints, *patches and updates* for pushing patches and software updates, and finally, *configuration change* for performing specific configuration changes at a system or application level. An expanded service definition for software services includes requirements for deployment (e.g., capacity requirements and minimum or maximum of computing resources such as CPU, disk, and memory), architecture requirements (e.g., platform and CPU specifications); prerequisites for deployment (e.g., software prerequisites for target endpoints and OS service packs); capabilities provided by service; and deployment artifacts such as process-automation workflows and images for OS installation.

The requirements, prerequisites, and capabilities are expressed in terms of asset attributes captured by the asset repository and the CMDB. This allows for the service catalog to perform matching of requirements with respect to capabilities of assets in the asset repository and ensuring that the requirements are met. For instance, a software application service that installs on the x86 platform and has a database server as prerequisite would express its requirements in a form such as (*hw.platform* == 'x86' and *database.component* == '*server*'). Note that the latter requirement is satisfied by a database server service in the service catalog that expresses its capability using exactly the same term. When a user selects this software package service from the service catalog, the necessary checks are performed to match these requirements and ensure that the software package is deployed on valid endpoints. Capturing dependency between multiple services using requirements and capabilities not only ensures successful deployment but also allows for preparing the environment by scanning the dependency chain and automatically provisioning requisite services. For example, for the above software application, if no asset containing a database server is readily available, then an appropriate endpoint is selected and the database server service is provisioned.

The client portal allows selected services to be added to a shopping cart. The end-user is also presented with valid assets on which these services can be deployed. On checkout, the purchase or request for provisioning is submitted to the system. On coDesk, the request flows through the various approvals before the actual deployment is initiated. The deployment is tracked, and on successful deployment, the asset repository is updated to create assets for the deployed components. If a ticket or incident request was raised for the service deployment, its status is automatically updated to reflect the successful deployment. The assets created in the asset repository can now be managed, and any configuration change service can be applied to it. **Figure 3** shows a screen capture of the client portal with a list of software installation items in the service catalog and matching assets.

The service catalog is integrated with embedded Tivoli Provisioning Manager (embedded TPM, or eTPM) [10] at the customer's end to perform the actual deployment. The CMDB entity records the various attributes of eTPM such as endpoint address and administrator credentials and maintains the relationship between the eTPM entity and the managed endpoints.

The deployment is performed by process workflows in the eTPM environment. TPM supports standard packaging and installation of software packages using software resource templates (SRTs) and standard workflows. A single SRT can provide all necessary information to deploy the software module in multiple platforms and different configurations. Workflows accept parameters and customize the installation according to the needs. These input parameters are optionally listed in the SRT along with their default values. The service catalog in coDesk imports SRTs and populates the service catalog schema. When a service is requested from coDesk, end-users have an option to proceed with the default parameters or specify their own to override the defaults.

Release management and configuration management processes

Some of the goals associated with ERMIS release management include the plan and oversight of successful provisioning of software and related hardware, and the design and implementation of efficient procedures for the distribution and installation of changes to IT systems, ensuring that the changes made are traceable and secure. The key stages of release management as deployed through a service catalog are as follows.

In the *build and configure* stage, after the user requests a particular service listed in the service catalog, the

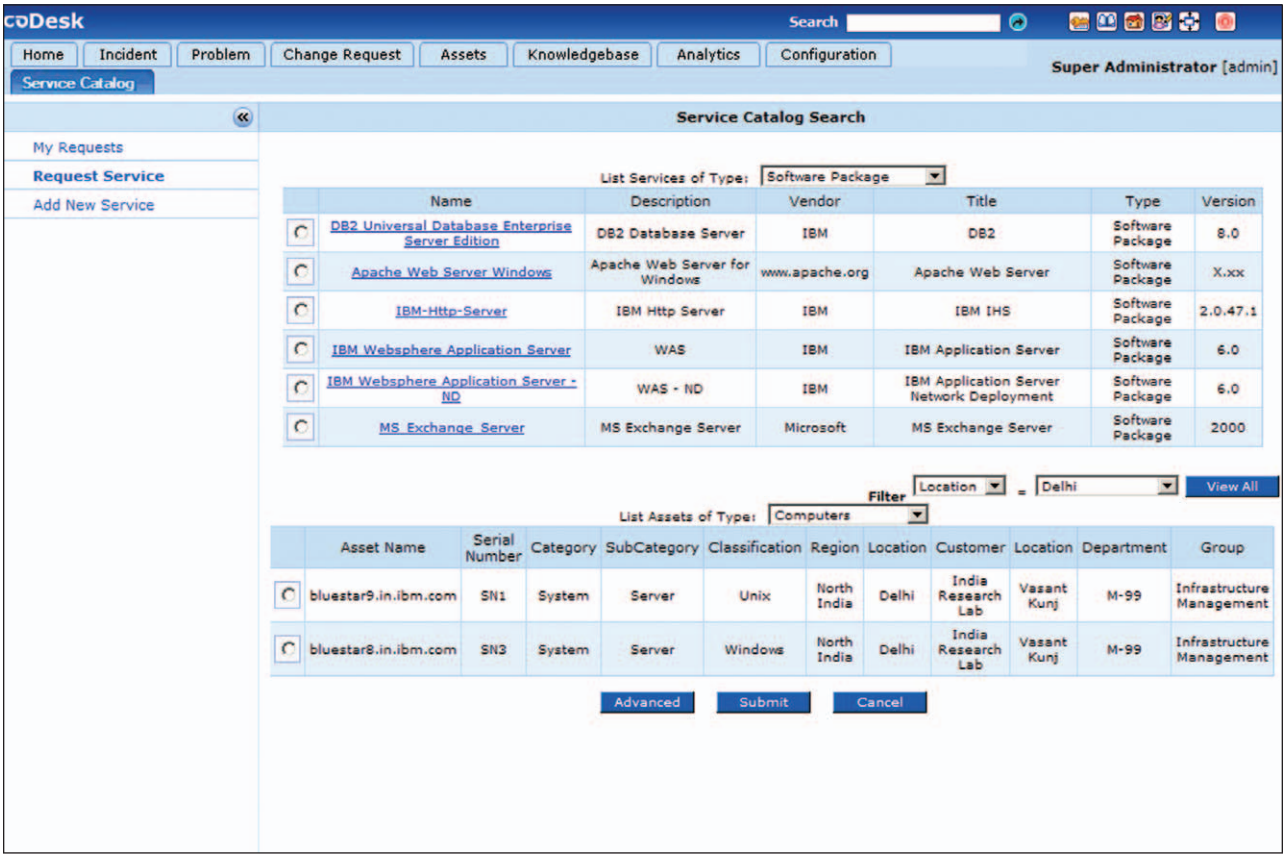

Figure 3

Screen capture of the service catalog in coDesk.

corresponding installers, configurations, and workflows are moved to a server that functions as a test bed. The requested software is installed, configured, and tested in the test bed (i.e., platform for experimentation) without affecting the other components on the network. The *test and accept* stage involves testing of the solution. Only after the installation has been thoroughly tested by a team is the software allowed to be installed.

In the *schedule and plan* stage, the appropriate time for the actual software installation in the production environment is determined along with the components that will be affected. In the *communication and preparation* stage, once the schedule and plan have been prepared, these need to be communicated to all of the affected people in the organization. The service in question may affect a number of systems and processes, and it is essential for the concerned people to be aware of and prepared for the same.

In the *distribute and install* stage, at the scheduled time, the required installers, configurations, and workflows are transferred to the target machine and the workflow executed. The workflow is a well-structured sequence of instructions to be executed on the target machine to prepare, install, and configure the installed software.

The *update asset database* stage represents the final step for software distribution, and it involves updating the asset database of the relevant entries. The CMDB in the ERMIS Asset Manager is updated to reflect the changes that were made to the systems and software. This also ensures the integrity of the data contained in the CMDB.

Software library with Asset Manager

ERMIS provides the software library management module to help our customers manage the software within the organization, including optimizing the utilization of software as well as the deployment, updating, and configuration of the software. The software library is an important component of Asset Manager, which stores all the software, description information, and the automation package to deploy and configure software within ERMIS. Each entry in the software library is called a *software module*. A software module stores all of

the information about the software to automate its deployment and configuration, such as name, version, vendor, the file used to install it, requirements to deploy the software, and the template to customize the deployment. To import software to the library, a software module based on the schema defined by ERMIS is written and then imported to the software library.

A software module includes the following parts: a software installation image including all the packages needed to install the software; prerequisites to deploy the software (e.g., OS and patch level), and a software template that predefines software configuration information. During the deployment period, the installation script will automatically extract these parameters from the template and use the template to customize the installation process. The software template is very similar to a response file that is needed in a "silent" install mode. The software library module also provides an administration console to manage the software. The user can query, add, update, and remove software from the software library as needed. The TPM enhances ERMIS in order to manage the complex software library and distribute software from the NOC to the customer site.

Software image synchronization with client software library

Software synchronization is required to link different sites and to synchronize the images between them. The architecture of the system is shown in **Figure 4**. The components of this architecture include 1) a master repository of software images at the NOC (lower right in Figure 4); 2) distributed repositories that act as cache and provide a local presence for faster downloads; and 3) a local repository at the customer site for temporary storage and distribution within the site (lower left). Because of the distributed nature of this architecture, there is a need for a synchronization scheme to keep the global and local repositories synchronized.

The software synchronization module contains an agent server placed in a customer site to synchronize the software between the ERMIS NOC and the customer sites. The agent server is composed of the following components: 1) a task manager that accepts or fetches tasks from the NOC and triggers the image synchronization task, 2) an image downloader that downloads images from one or multiple depot servers, 3) a local repository that saves downloaded images on a local disk, and 4) an installation engine that executes installation scripts on target servers.

During the synchronization process, the agent server receives software images from multiple depot servers. First, the ERMIS NOC server distributes the software images across multiple depot servers. Next, the NOC operator initiates a software synchronization task by sending a start task command to the agent server in the customer site. This is followed by the agent server checking the image availability and obtaining pieces of images (in a manner similar to BitTorrent** processes) from various repositories that are globally distributed (and even from other customers who may already possess copies), while giving preference to the repository that is the closest and *not* from the ERMIS NOC. After the images are successfully downloaded, the agent server registers the software in the customer's software library. This approach dramatically reduces traffic from the ERMIS NOC while transferring the load closer to the customer sites, which in turn radically increases image distribution speed. This approach also serves the second important goal of retaining central control over all software synchronization tasks at the NOC.

Software deployment

Once the software has been distributed to the closest locations to the customers, the important task of software deployment begins. Embedded TPM is used to deploy software to the target servers; thus, TPM must be preinstalled on the agent server in the target environment. The agent server receives the task from the NOC site and executes it locally in the customer's environment. Tasks include configuring the servers and deploying software at the customer site. The software deployment process checks whether the image and automation package are available at the customer site. If not, it is distributed and installed at the customer site, as described in the previous section. The NOC server is now ready to deploy the software. First, the server initiates the software deployment process. Next, the agent server receives the command, parses it, and invokes TPM in order to deploy. Finally, the agent server obtains the provisioning status and sends it back to the NOC. If the deployment process fails, the NOC will receive the error message from the agent server.

Problem determination and resolution

PDR is the process of detecting anomalies in a monitored system, locating the problems responsible for the failure, and fixing the cause of the problem. The main focus of this discussion is the enhancement of the current remote monitoring infrastructure, SNAPPiMON, which provides anomaly detection, with problem classification automation, integration with the IBM Support Assistant (ISA) [11], and a probe-generating framework.

Anomaly detection in the managed environment

Currently, SNAPPiMON provides visibility, insight, and control with respect to monitored systems, enabling fast detection and notification of IT infrastructure anomalies.

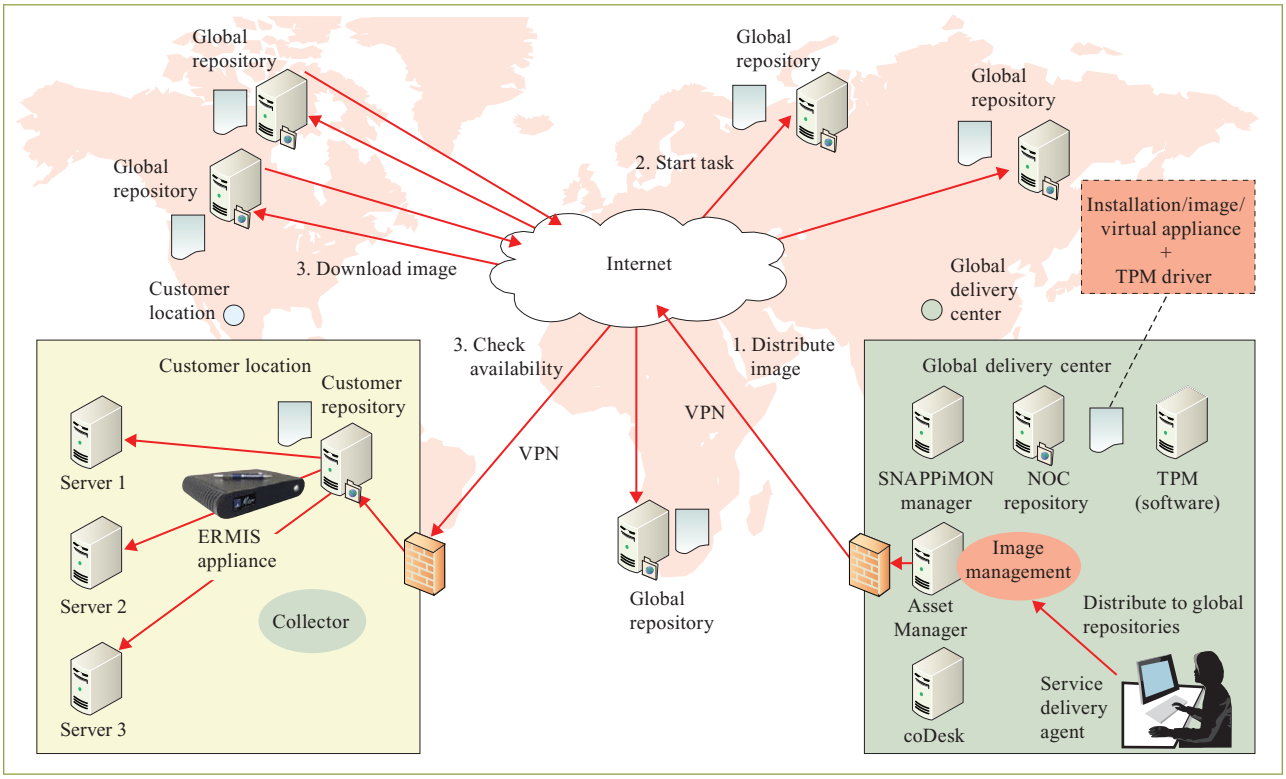

Figure 4

Software synchronization between different sites. The numbers 1, 2, 3, and 4 toward the center of the figure indicate steps in the synchronization process.

Using *IT service views*, users across an organization are able to monitor a large spectrum of IT metrics and track IT service levels. SNAPPiMON automatically provides an alert when thresholds are breached by displaying the corresponding events on the dashboard. However, in the case of problems in complex IT environments with multi-tier system dependencies, the same issue can cause failures that generate multiple notifications at different levels of the multi-tier system. In such cases, the end-user will benefit from embedded mechanisms that classify the issues experienced by the failing resource, collect relevant data to localize the problem at hand, and then generate a descriptor that can be used in the next occurrence of the problem.

Problem classification automation using the problem pattern learning tool

The first step is to build an automation framework that extends the remote-monitoring SNAPPiMON infrastructure functionality by classifying the problems detected in a predefined hierarchical structure of taxonomies (**Figure 5**). We use an incremental online learning algorithm that learns the pattern of errors from a set of manually labeled training data.

When a problem occurs, either the system or application will generate some error logs or the user or administrator will notice alarms or abnormal changes in performance of the system. SNAPPiMON (center of Figure 5) collects error log data and system performance data. These two types of data are combined for a more powerful feature set for problem classification. The log data contains word features, while the performance data are numerical values. These two types of data are uncorrelated. In the field of machine learning, it has been shown that combining features that are less correlated will generally exhibit better predictive performance for the model [12].

To automate the process of problem classification, training data is needed for the underlying classifiers so they can learn the patterns of problem taxonomy. This requires the manual labeling of each problem instance. In this problem setting, each problem instance contains a set of labels that is referred to as a *multilabel* (e.g., {MySQL, general}, {WebSphere, runtime, CPU}). A multilabel is a set of labels that includes the name of the application or

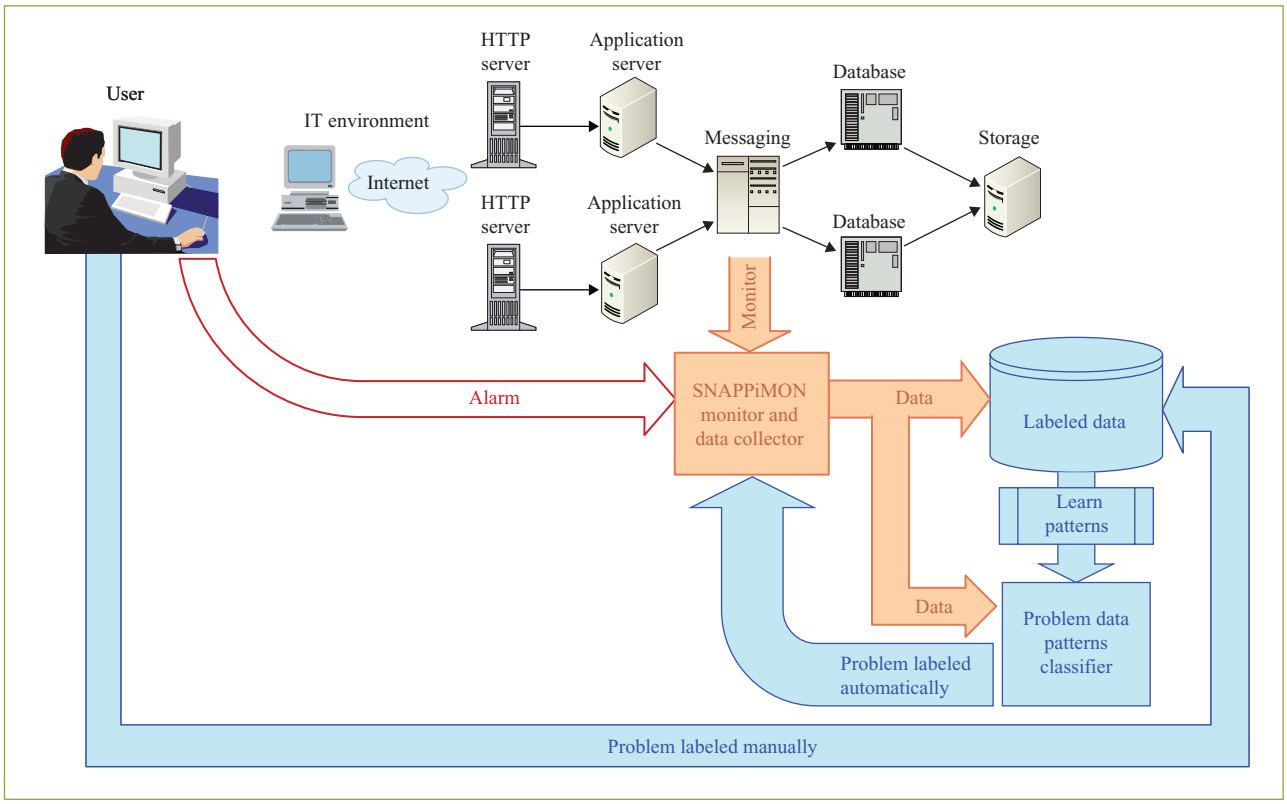

Figure 5

ERMIS monitoring with automatic problem classification (in blue). (HTTP: Hypertext Transfer Protocol).

product, as well as the specific cause of the problem. The set of labels can be represented using a hierarchical forest structure, where each tree in the forest specifies possible problem causes for a specific application or product. The name of the application or product is assigned to the root of the tree whose level structure is decided by the complexity of the application or product in question. Regardless of the derived tree structure, the path of the labels always starts at a root node and ends at a leaf node. For each individual problem, there is one and only one such path in the entire forest. Note that whenever a certain node is included in the label set, all of the nodes on the path to the root node must also be included. We represent the multilabel using a binary coding schema, where 1 indicates the presence of the problem instance at a node, and 0 indicates the absence of the problem instance at a node. In the example shown in **Figure 6**, where a total of $M = 10$ labels exist, the problem instance (in red nodes) has two labels: labels 1 and 3. Thus, the associated multilabel can be represented as {1, 0, 1, 0, 0, 0, 0, 0, 0, 0}.

By transforming the structure of a flat problem space into a hierarchy that is predefined according to an existing taxonomy, the complexity of the pattern search for PD is reduced. For the purposes of the ERMIS prototype, we make use of the problem taxonomy provided by IBM software brands for their software serviceability workbench ISA [7].

To learn the patterns of the problem taxonomy, and to train the classifiers, we use a linear classifier at each node in the hierarchy. Linear classifiers exhibit better generalization toward large sets of data, avoid the overfitting problem since they require less internal parameters to learn, and thus are scalable in enterprise environments. There are many choices of linear classifiers, including discriminant classifiers such as Fisher's linear discriminant, and generative classifiers such as naive Bayes. We use the Perceptron algorithm [13] for our online learning purpose because the algorithm can incrementally update the internal parameters whenever a new training instance becomes available. Compared to the traditional batch learning algorithms, this online learning framework can significantly decrease the computational complexity of the training process by learning from new instances in an incremental fashion. At the same time, this reduces the amount of memory

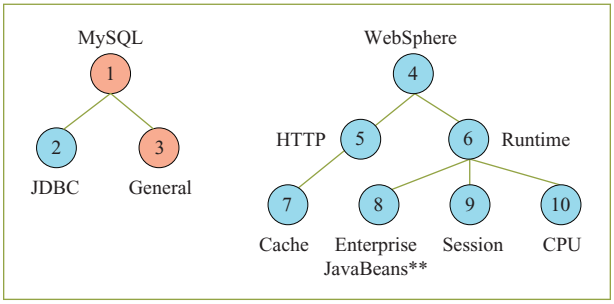

Figure 6

Multilabel example.

required to store the training instances. The details of the algorithms are discussed in Reference [14].

After training the classifiers, each node stores only the internal parameters in the memory. The trained classifiers can be used for real-time PD when a new problem is reported by the ERMIS monitoring tool. Specifically, when a new problem occurs, X, the multilabel, is computed automatically using the same process used for training, except for the condition that a node is visited only if its parent node classifies the instance X to be positive.

We evaluated the performance of four algorithms by measuring both their *effectiveness* and *efficiency*. The effectiveness is calculated by using the *precision* score, i.e., the percentage of correctly classified data in the entire test dataset. To evaluate the efficiency, we consider the computational complexity for training the classifiers. We used Trade 6 [15] as a test bed for Web applications. For simulating user operations, we used IBM WebSphere Workload Simulator [16] to perform multiuser activities on Trade 6. Four types of errors were injected in order to generate errors so that we could collect the training data, i.e., errors associated with database shutdown, network failure, WebSphere port change, and user connection-limit exceeded. Both log data and system performance data were collected. Overall, a total of 1,700 error samples were collected, approximately 2,800 word features, and 80 numerical features. With the combination of both log data and numerical data, our algorithm performs best, compared to log data and numerical data alone, in all three cases where the training data contains 10%, 50%, and 90% of the entire data.

With the combination of log and numerical data, our algorithm achieves a 96.35% precision with 90% of the training data.

Since the problem taxonomy used for our prototype is the one provided by ISA, our next logical step links SNAPPiMON and ISA.

Integration with ISA serviceability workbench

ISA, or the IBM Support Assistant, is an IBM serviceability workbench for software products that allows the users to 1) find relevant information by federating multiple search engines, 2) analyze the problem, and 3) manage a problem request. The main goal of the ISA is to accelerate the problem resolution for customers. ISA contains a predefined problem taxonomy for IBM products. Currently, in order to invoke a search or problem analysis via its specialized data collectors (one for each problem in the ISA taxonomy), customers must know the problems they are dealing with. In case of software problems in a multi-tier IT environment with complex distributed system dependencies, the same front-end issue may have its origins in different back-end system or application problems. Thus, the fixes found for the symptoms initially discovered may not be addressing the actual root cause.

An example of such a multi-tier environment is an e-business system supported by an infrastructure consisting of the following subsystems connected by local area networks and WANs: Web-based presentation services, access services, application business logic, messaging services, database services, and storage subsystems.

The objective of the enhanced ERMIS prototype discussed here is to automate the problem classification by programmatically matching problem data to learned patterns according to the ISA problem taxonomy. Thus, as we illustrate in **Figure 7**, when a problem is reported by SNAPPiMON, the user, or the administrator, the $DATA$ monitored and collected by SNAPPiMON is used to classify the problem according to the ISA taxonomy as outlined earlier. Once the type of error is known, the corresponding ISA collector is invoked to select the relevant subset $DATA'$ from the entire SNAPPiMON $DATA$ to be analyzed by ISA for problem diagnosis. Hence, by combining the enhanced SNAPPiMON and ISA, we extend SNAPPiMON capabilities into problem diagnosis.

Probe-generating platform

Finally, we propose a probe-generating platform to further facilitate the problem localization in a more productive manner. The basis of this approach is to build tools for developing probes descriptors and runtimes for executing them in a problem context. The effort is focused on reusing the probe descriptors (developed by experts) in various contexts, across managed environments, without manual interventions.

Modeling tools to create descriptors for probes and diagnosis tools

In order to locate the cause of a problem, administrators often need to execute various probes on a distributed

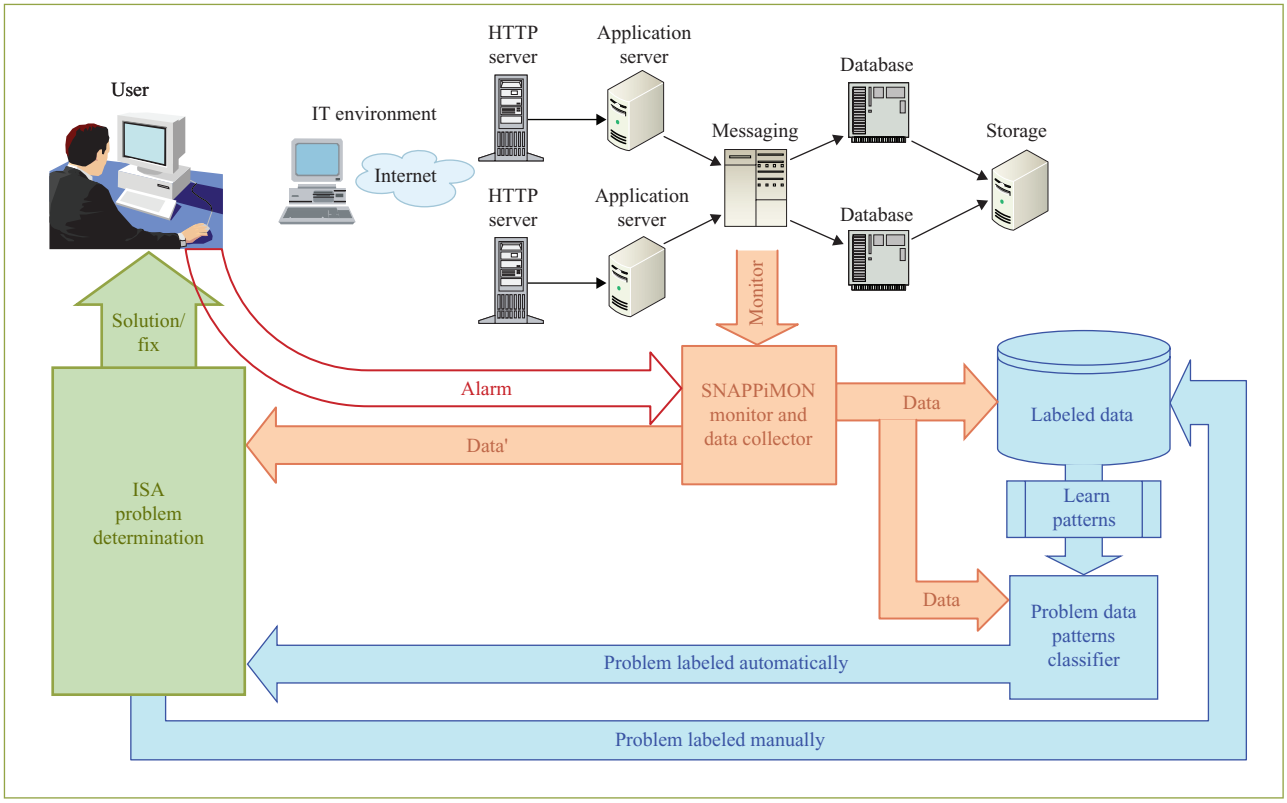

Figure 7

Problem diagnosis using the enhanced ERMIS Monitoring (in blue) and ISA (IBM service agent) in green.

system. Typically, these probes cannot be executed all the time, unlike monitoring probes, due to the associated overheads. Two examples of such system probes are "telnetting" to a database server port or invoking an operation of an application server via Java Management Extension (JMX) [17]. The telnet probe can be used to check whether a database server is running, while the JMX probe can be used to check whether the connections held by the application server are usable.

Traditionally, the PD activities in a complex IT environment are performed mostly in an ad hoc manner by various experts (e.g., database or network administrators or support teams) executing specialized probes and interpreting the outputs in a collaborative manner to localize the issue. There are three main challenges with respect to this probe-based approach to problem localization, which increases the time for problem resolution and hence has an impact on the productivity. First, the probes are very specialized and require experts (for setting the right execution environment, deciding the parameters for a given context, executing a probe, and interpreting the output). Second, the probes (e.g., scripts) are often highly customized for a specific environment and are difficult to reuse, even among experts. An example of customization is the topology assumed for the application components. Finally, transitions from one probe to another cannot be performed in a systematic manner, particularly if the probes belong to different technology domains.

In order to cope with these limitations and reduce the time to localize a problem, we use the fact that although in an enterprise IT environment the number of applications is large, the number of different products employed can be small. Further, most of the probes supported by products remain the same across environments, and the changes to these probes are typically limited to the environment-specific configurations that can be obtained from the CMDB. We are building a framework that uses the principles of service-oriented architecture (SOA) and information integration to automate PD activities. Following SOA principles, various reusable activities are modeled as services that are combined to offer higher-level services. Examples of such services include probe execution services (e.g., a script execution service that uses as input a script, the values of associated parameters, and the

execution environment); output analysis services (e.g., a service that parses the text output of a probe execution service and creates output in the form of structured data, given mapping rules, and possibly based on regular expressions); and information integration services (e.g., a service that provides values for variables given the mapping from query to information stores such as CMDB).

For example, a simple service template executes an OS *command* and produces *textOutput* that contains the output of the command. This service executes the command on its hosting machine. An advanced service for a similar purpose may use *host*, *sshUser*, and *sshPassword* as other possible inputs for executing the *command* remotely via ssh (secure shell). Another example is a service template that creates an instance of a Java class, namely *javaClassName*, and populates its attributes using the regular-expression-based functions applied on text, *attribute_RegEx*, which contains the mapping of attribute names to such functions.

Modeling tools to automate repetitive PD processes

A high-level probe can be described as a composition of various lower level probes; for example, an *IsDBServerUP* probe may be composed of a *Telnet Probe*, a *text analysis probe* (to analyze the telnet output), and a *CMDB Query Service*. This probe provides a higher level of abstraction by using *DBServerID* (a unique ID of a DBServer in the CMDB) as input and producing a Boolean output, namely *isDBServerUP*, that shows whether the DBServer is running well. When a composition of probes also includes a control flow logic, the higher-level probe is referred to as a *probe plan*. Typically, probe plans generated in this manner are the foundation of our repetitive PD processes.

The integration of the technologies outlined above, including problem classification, probes, and ISA, using the ERMIS monitoring platform, lead to cost savings and a PDR duration decrease. It saves labor for building and maintaining environment-specific scripts and probes. It also enables reuse of PD plans across accounts and provides the abstraction of probe knowledge for other people to compose higher-level probes. It empowers the service desk operators to execute relevant PD plans to gather better failure targeted monitoring data. Thus, the SNAPPiMON monitoring *DATA* is filtered on the basis of the PD plans resulting in *DATA′* for ISA for further analysis before forwarding the problem tickets to highly skilled technical support personnel.

Conclusion and future work

We have presented a new remote management service offering called Express Remote Managed Infrastructure Services (ERMIS). Since this class of offerings provides an opening into the SMB market for IBM, it was imperative to build the differential artifacts to lower costs and enhance the global appeal and reach of RIM via automation. To that end, we described two areas of research: service catalog and PD. Automation of management tasks for such commonly requested services helps in lowering costs while quickly delivering leading-edge services to customers.

The ERMIS service catalog aids in providing functionality across different ITIL processes such as service catalog management and service-level management, as well as release, configuration, and change management. Services can have relationships with other services, and services may have a dependency on or prerequisite relationship with other services. Further, a service can be composed of multiple other services and can also be bundled. A provisioning appliance is utilized at the customer site to act as an agent for server discovery and software installation. This approach uses the concept of service classes, which provide discrete configuration points, embedded in the service catalog. Monitoring data from the NOC-I toolset is used to periodically learn and provide end-to-end solutions that more accurately match the user's service requirements.

As software solutions become more complex, it becomes impossible for the administrator of the data center to master the skill of each solution and learn the complex dependency relationships among the components in different solutions. In the ERMIS context, future work is envisioned that deploys a virtual appliance 1) using streaming technology to decrease the virtual image transformation time [18], 2) deploying virtual images to a shared network storage (virtual machine file system or network file system) instead of the local storage of the hypervisor to support active migration [19], 3) redeploying of virtual images [20], and 4) using patch deployment of many homogenous virtual images.

The primary challenge associated with PD and resolution tasks was to effectively integrate them into the available ERMIS NOC-I remote monitoring infrastructure. Probe-generating techniques were used for better targeting of the problem location. Knowledge management assists with problem diagnosis and analysis by using classification and machine learning techniques. We also provided integration with existing tools that guide the service request submissions.

Future work includes research in the area of the PD 2.0 framework to refine how we allow people to build PD plans in the laboratory and deploy them in a customer environment (or even a data center) without requiring changes in those plans, instead of automatically customizing them based on CMDB data. Thus, the entire community can contribute with various low-level services, simple probes, composite probes, or probing plans to a

shared repository. Experts may also develop various PD plans in their laboratories and share them for reuse in any data center or RIM environment without requiring changes to the PD plan template.

The architectural decisions in our ERMIS project were largely driven by the considerations and requirements of this emerging business that aims to serve small and medium businesses. From the design of the tools to the deployment model, the main focus of this project has been to create a user-friendly, highly scalable solution that is affordable and consumable in the SMB space.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of the U.K. Office of Government Commerce, Linus Torvalds, Sun Microsystems, Inc., Oracle International Corporation, Sybase, Inc., MySQL AB, Microsoft Corporation, The Open Group, Google, Inc., or BitTorrent, Inc., in the United States, other countries, or both.

References

1. Patni Computer Systems Limited, "Remote Infrastructure Management"; see *http://www.patni.com/infrastructure-management/service-offerings/remote-managed-services.aspx*.
2. Hewlett-Packard, "Reducing the Complexity of IT Environments," white paper; see *ftp://ftp.hp.com/pub/services/infrastructure/info/it_reducing_complexity.pdf*.
3. Accenture, "Infrastructure Solutions"; see *http://www.accenture.com/Global/Technology/Infrastructure_Solutions/default.htm*.
4. HCL Technologies Limited, "Services+"; see *http://www.hcltech.com/Services+/*.
5. Infosys Technologies Limited, "Infosys—IT Services"; see *http://www.infosys.com/IT-services/default.asp*.
6. IBM Corporation, SNAPPiMON Datasheet; see *http://www-07.ibm.com/in/noci/downloads/it_monitoring.pdf*.
7. IBM Corporation, coDesk Datasheet; see *http://www-07.ibm.com/in/noci/downloads/nov16_codesk3.0.2_data_sheet.pdf*.
8. IBM Corporation, Asset Manager Datasheet; see *http://www-07.ibm.com/in/noci/downloads/it_assetmgmt.pdf*.
9. Cisco Systems, Inc., "Internetworking Technology Handbook: Simple Network Management Protocol"; see *http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html*.
10. IBM Corporation, "Tivoli Provisioning Manager Documentation"; see *http://publib.boulder.ibm.com/infocenter/tivihelp/v16r1/index.jsp*.
11. IBM Corporation, "IBM Support Assistant"; see *http://www-01.ibm.com/software/support/isa/*.
12. I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.* **3**, 1157–1182 (2003).
13. R. Rojas, *Neural Networks: A Systematic Introduction*, Springer-Verlag, New York, 1996.
14. Y. Song, A. Sailer, and H. Shaikh, "Problem Classification Method to Enhance the ITIL Incident, Problem and Change Management Process," *Eleventh IFIP/IEEE International Symposium on Integrated Network Management, IM 2009*, June 2009.
15. IBM Corporation, "IBM Trade Performance Benchmark"; see *https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=trade6*.
16. IBM Corporation, "WebSphere Studio Workload Simulator for z/OS and OS/390"; see *http://www-306.ibm.com/software/awdtools/studioworkloadsimulator/*.
17. Sun Microsystems, "The Java Community Process Program"; see *http://jcp.org/en/home/index*.
18. L. Shi, M. Banikazemi, and Q. B. Wang, "Iceberg: An Image Streamer for Space and Time Efficient Provisioning of Virtual Machines," *International Conference on Parallel Processing*, September 8–12, 2008, pp. 31–38.
19. C. Sun, L. H. Q. Wang, and R. Willenborg, "Simplifying Service Deployment with Virtual Appliances," *Services Computing, 2008*, Honolulu, HI, July 7–11, 2008, pp. 265–272.
20. VMware, "Open Virtualization Format (OVF)"; see *http://www.vmware.com/appliances/learn/ovf.html*.

Received December 17, 2008; accepted for publication March 3, 2009

Mahesh Viswanathan *IBM Global Technology Services, 294 Route 100, Somers, New York 10589 (maheshv@us.ibm.com)*. Dr. Viswanathan is a Senior Technical Staff Member in IBM Global Technology Services. He is Chief Architect for the Express Remote Managed Infrastructure Services (ERMIS), a managed services product for remote delivery of infrastructure services. Dr. Viswanathan was a Research Staff Member at the IBM Thomas J. Watson Research Center. His interests include remote services delivery, service-enabled information systems, high availability of composite applications, human machine interfaces in cars, speech recognition and synthesis, audio/video search and retrieval, and document image analysis. He has a Ph.D. degree in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute, specializing in image analysis. He has more than 30 technical publications and 20 international patents and is an IBM Master Inventor and a senior member of the Institute of Electrical and Electronics Engineers (IEEE).

Hidayatullah Shaikh *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (hshaikh@us.ibm.com)*. Mr. Shaikh is a Senior Technical Staff Member and Manager at the IBM T. J. Watson Research Center. He leads and manages projects in the areas of remote IT infrastructure management and virtual clients. His areas of interest and expertise include virtualization, cloud computing, remote services delivery, business process modeling and integration, service-oriented architecture, grid computing, e-commerce, enterprise Java, database management systems, and high-availability clusters.

Anca Sailer *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (ancas@us.ibm.com)*. Dr. Sailer received her Ph.D. degree in computer science from Pierre et Marie Curie University of Paris, France, in 2000. She was a Research Member at the Networking Research Laboratory at Bell Labs from 2001 to 2003, where she specialized in Internet services traffic engineering and monitoring. In 2003, Dr. Sailer joined IBM where she is currently a Research Staff Member in the Service Products department. Her interests include cloud computing management, self-healing technologies, and data analysis.

Yang Song *Microsoft Research, One Microsoft Way, Redmond, Washington 98034 (yangsong@microsoft.com)*. In 2008, Dr. Song received his Ph.D. degree in computer science and engineering from Pennsylvania State University. His research interests include developing and applying statistical machine learning methods for text classification and clustering, pattern recognition, Web information retrieval, and data mining. Dr. Song has a B.E. degree in computer science from Zhejiang University, China. In 2009, he

joined Microsoft Research where he is currently a Research Member in the Search Quality Group.

Xing Fang *IBM China Investment Company Limited, IBM Global Service China Company, Tower B, Building 19, Zhong Guang Cun Software Park, 8 Dongbeiwang WestRoad, Haidian District, Beijing 100193, China (xingfang@cn.ibm.com)*. Mr. Fang received B.S. and M.S. degrees in computer science from Tsinghua University, China, in 2001 and 2004, respectively. He joined the IBM China Research Laboratory in 2004 where he worked for four years. He participated in several research projects concerning cloud computing, performance optimization, and SOA. He has ten patents. He now works as an Architect in the IBM Cloud Computing Center in China.

Yu Hui Wu *IBM China Investment Company Limited, Zhong Guang Cun Software Park, Haidian District, Beijing 11 100193, China (wuyuhui@cn.ibm.com)*. In 2007, Mr. Wu received his M.S. degree in computer science and technology from Harbin Institute of Technology. He is currently a Research Member of the Distributed Computing and Service Management team at the IBM China Research Laboratory. His interests include the virtualization and provisioning technologies used in cloud computing.

Zhi Le Zou *IBM China Investment Company Limited, IBM Global Service China Company, Tower B, Building 19, Zhong Guang Cun Software Park, 8 Dongbeiwang WestRoad, Haidian District, Beijing 100193, China (zouzhile@cn.ibm.com)*. Mr. Zou received his B.S. and M.S. degrees in Computer Science from Xidian University, China. Since 2007, he has been a researcher at the IBM China Research Laboratory. His major interests are virtualization and cloud computing technologies.

Kishore P. Reddy *JaMocha Tech Private Limited, #730, 2nd Floor, 3rd Block, Koramangala, Bangalore 560034, India (kishorepr@jamochatech.com)*. Mr. Reddy previously led the engineering efforts for NOC-Inside services as part of the Software Group at Network Solutions, an IBM Company. He left this company to pursue his entrepreneurial interests. As Head of Engineering at JaMocha Tech, he continues to work in the areas of end-to-end infrastructure management covering both IT and non-IT assets. He is also involved in exploration of areas concerning energy management and telemetry.

Abhijit Deshmukh *Network Solutions Pvt. Ltd., 32 Grape Garden, 17th H Main Road, 6th Block, Koramangala, Bangalore 560095, India (abhideshmukh@in.ibm.com)*. Mr. Deshmukh holds a Master's degree in software engineering from Rensselaer Polytechnic Institute. He has more than 14 years of experience in product and services engineering and technology and software development. At Network Solutions, an IBM Company, he is responsible for business and engineering of NOC-Inside services for automated IT infrastructure management.

Manish Gupta *IBM India Pvt. Limited, India Research Laboratories, Plot 4, Block C, Institutional Area, Vasant Kunj, New Delhi 110070, India (gmanish@in.ibm.com)*. Dr. Gupta is a Research Staff Member at the IBM India Research Laboratories, Delhi, where, since 1998, he has been actively involved in e-commerce and systems management research. He has developed reverse auctions for the IBM WebSphere Commerce Suite software, led the design for the health management component of WebSphere XD v6.1 (IBM quality-of-service management middleware). His areas of interest are in autonomic computing, services-oriented architecture, and Web 2.0. He completed a Ph.D. and Master's degree program in computer science from Indian Institute of Science, Bangalore, in 1998 and 1994, respectively.

Bharat Krishnamurthy *IBM India Research Laboratories, Block D, EGL, Bangalore, India (bharat.kris@in.ibm.com)*. Mr. Krishnamurthy is a Technical Staff Member in the Infrastructure and Resource Optimization group at the IBM India Research Laboratories. He joined IBM in 2006 and has been working in the areas of infrastructure monitoring and systems management. Prior to joining IBM, he was working as a Senior Engineer at Thinix, Inc., where he designed hybrid clients and client management appliances. He holds a Master's degree in computer science from the University of New Mexico.

Manish Sethi *IBM India Pvt. Limited, Plot 4, Vasant Kunj Phase Ii, Vasant Vihar Institutional Area, New Delhi DL 110070, India (manishsethi@in.ibm.com)*. Mr. Sethi is a Technical Staff Member at the IBM India Research Laboratories. He joined IBM in 2004 and his broad areas of interest include middleware and systems management. In the past, Mr. Sethi was a member of the JSR-160 expert group for standardizing JMX remote application programming interfaces. He holds a Master's degree in computer science from Delhi College of Engineering.

Balaji Viswanathan *IBM Research Division, India Research Laboratories, EGL, Bangalore, India (bviswana@in.ibm.com)*. Mr. Viswanathan has been with IBM since 2004 and is part of the Infrastructure and Resource Optimization group. His focus areas are systems management, remote deployment and provisioning, grid, and workflow systems. His interests include Linux, virtualization, and mobile computing. He has an M.S. degree from Pennsylvania State University.

Joseph G. Gulla *IBM Global Technology Services, 3039 East Cornwallis Road, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (gulla@us.ibm.com)*. Dr. Gulla is an Executive IT Specialist for IBM Global Technologies Services. His focus is in the development and support of server-based service products for the global sales community and their customers. Previously, he worked on managed hosting service offerings in Europe and Asia. His research interests include collaboration, systems development process, applications management, and systems and network management. Dr. Gulla is also a faculty member of Nova Southeastern University where he teaches distributed computing, enterprise architecture, and systems development process courses. Dr. Gulla's dissertation, titled "Design and Implementation of a Prototype Toolset for Full Life-Cycle Management of Web-Based Applications," was published as an IBM Technical Report in 2003.

Fouad Matar *IBM Global Technology Services, Services Delivery, 800 North Magnolia Avenue, Orlando, Florida 32803 (matar@us.ibm.com)*. Mr. Matar is a Senior Technical Staff Member in the Global Services Delivery organization, currently on assignment as the Services Delivery Leader in the Middle East and North Africa Growth Market Region. Having supported commercial engagements as an architect and technical solution manager, and more recently participated in the advancement of strong linkages within the Global Technology Services product lines and IT Delivery teams, Mr. Matar is now focused on applying this experience as the delivery leader in the growth markets. He has an M.S. degree in electrical engineering from the University of Central Florida and is currently working on a Ph.D. degree in mathematics.