

Autonomic Computing Strategy Perspectives

3

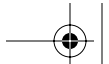


This chapter will address one of the fundamental on demand business strategy perspectives, *Autonomic Computing*. This chapter's focus on Autonomic Computing will emphasize both strategic and technological perspectives. The overall on demand business strategy focuses on two computing areas of interest: *Autonomic Computing* and *Grid Computing*. It is the Autonomic Computing disciplines that provide the necessary efficiencies required to conduct on demand business.

Previously, we discussed the need for critical intersections to occur among IT, business operations, and autonomic transformations. We introduced the three high-level aspects of on demand business and the concepts driving these types of environments. In this chapter, we will provide further treatment of the strategic perspectives defining the on demand business world of operations, including the IBM Autonomic Computing strategy.

Here is another example of Autonomic Computing: A computer intermittently freezes up. No customer transactions can be processed for several seconds, costing thousands of dollars in business and customer loyalty. Today, the IT support staff might not even find out about the problem for more than





a day; and when they do, it may take a couple days to figure out what the problem is and which of the multiple scenarios matches the problem.

With an autonomic system in place—with real-time event monitoring and auto-tuning analysis—the freeze-up is detected the very first time it happens and matched against historical problem data. The settings are reset automatically, averting any real loss of revenue and customer loyalty. A report then shows the actions that were taken to resolve the issues.

And yet another example of Autonomic Computing: If an airline starts a fare sale and is hit by a high volume of customer inquiries, it would take less than a minute for the autonomic software to determine that more power is required and add another computer. The system can also turn off computers as they are no longer needed.

Autonomic Computing introduces *autonomic* efficiencies into the overall scheme; however, Autonomic Computing alone does not entirely constitute an on demand business. As we discussed in Chapter 1 of the book, on demand business involves three fundamental activities to transform to an efficient on demand Operating Environment.

ON DEMAND BUSINESS...

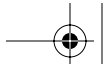
The IBM Corporation has defined *on demand business* as an enterprise whose business processes—integrated end-to-end across the company and with key partners, suppliers, and customers—can respond with agility and speed to any customer demand, market opportunity, or external threat.

Respective to the Autonomic Computing vision, are capabilities like maintaining security against undesirable intrusions, enabling fast automatic recovery following system crashes, and developing “standards” to ensure interoperability among myriad systems and devices. Systems should also be able to fix server failures, system freezes, and operating system crashes when they occur; or better yet, prevent them from developing in the first place.

An on demand business is a way of doing business (a strategy), and the on demand Operating Environment consists of the systems and procedures that are used to do business (executing the strategy). As a review of our previous discussion, and to set the stage for this discussion on Autonomic Computing, let us again consider these three key activities. Each activity is paramount to the successful transformation of any corporation, enterprise, or organization with a desire to be an on demand business. These three activities are:

1. Engaging in several “levels” of streamlining and transforming one’s overall business enterprise processes.





2. Embracing new mechanisms for the delivery of services and pricing, which is also referred to in the industry as “*Utility Computing*.”
3. Ensuring the enterprise is delivering and sustaining a flexible operating environment.

As described throughout much of this chapter by David Bartlett (IBM’s Director of Autonomic Computing), Autonomic Computing plays an important role in several on demand business activities. Several fundamental questions arise as one begins to consider what is involved in the on demand transformation, as illustrated Figure 3.1:

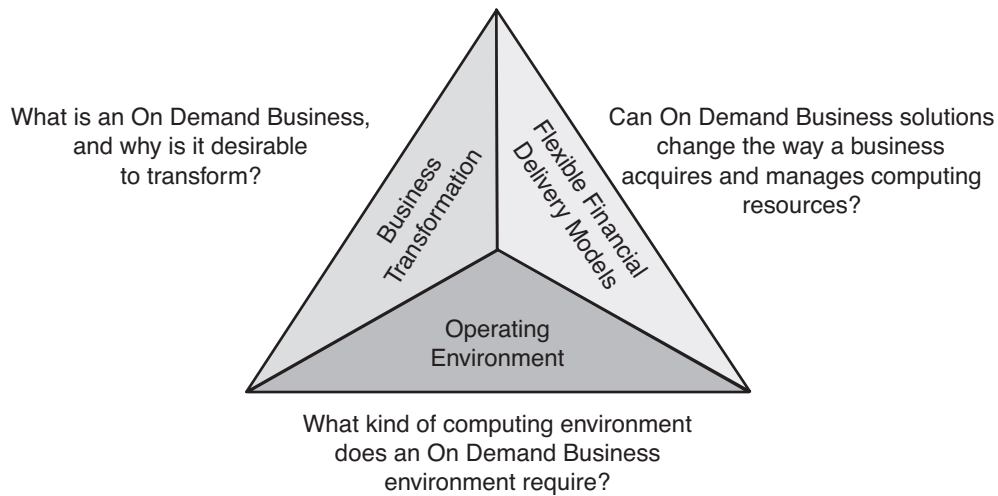
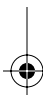
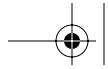


FIGURE 3.1 The key focus areas in Autonomic Computing, which must be addressed while creating an on demand business environment. The questions unveil critical thinking related to a business’ transformation.

Exploring these three key questions begins to unveil what is necessary to build an on demand Operating Environment. Autonomic Computing is all about business transformation, which requires engaging in several “levels” of streamlining and transforming one’s overall business enterprise processes. This will involve the shift of IT infrastructure from a reactive, human-intensive management paradigm to an adaptive, technology-balanced approach, supported by Autonomic Computing technologies. *Utility Computing* is also involved; this embraces new mechanisms for the delivery of services and pricing methods. The Utility Computing approach also involves the integration of attractive, new, flexible financial delivery models.

Let us explore a bit more the sections illustrated in the pyramid of Figure 3.1:





Business transformation involves transforming an organization's strategy, processes, technology, and culture by applying deep business process insight with some combination of advanced technologies to increase business productivity and enable flexible growth.

The *operating environment* is typically an approachable, adaptive, integrated, and reliable infrastructure for delivering on demand services to an on demand business operation.

Flexible financial and delivery offerings embrace the delivery of business processes, applications, and/or infrastructure On Demand, with usage-based charges around IT and/or business metrics. This is a new approach for most IT departments and service providers, either working independently or together. It is paramount to more effectively align IT assets with business priorities, provide reliability through more granular SLAs (Service Level Agreements), and achieve cost savings through increased utilization and proactive management.

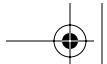
Autonomic Computing, with its focus on reducing management costs, leverages the Utility Computing approach by enabling infrastructure and on demand business process services to subscribing clients at the lowest possible cost. Autonomic Computing, quite obviously then, has the most obvious and direct connection to ensuring that one is delivering and sustaining a very flexible operating environment.

Consider the following question, which is being asked in Figure 3.1: What kind of computing environment does on demand require, and how do I build one? The answer will ultimately include functions and features such as the following: Monitoring, workload management, provisioning, dependency management, and policy-based computing. All these features are at the core of both the on demand Operating Environment and the Autonomic Computing framework.

Establishing the fundamental on demand business intersections for any company involves the implementation of standardized and automated processes, applications, and infrastructures over networks and advanced services with business and IT functionality. As published in the *Silicon Valley Business Ink* in April 2003 (by Alan Ganek [Ganek01], Vice President of Autonomic Computing for IBM Corporation's Software Group), the following article describes some innovative thinking and strategy perspectives regarding Autonomic Computing.

In this reprinted article, Alan Ganek states:





Technology is like a race car going 200 miles per hour on the fastest track on Earth. Systems crash. People make mistakes. Computers need a lot of maintenance to keep them up and running. This is life. At the same time, consumers have become increasingly intolerant of computer failures, while placing ever-greater demands on the technology they utilize.

The high-tech industry has spent decades creating systems of marvelous and ever-increasing complexity, but like Charlie Chaplin falling into the machine in "Modern Times," complexity itself is the problem.

Consider this fact: One-third to one-half of a company's total IT budget is spent preventing, or recovering from system crashes, according to a recent study by the University of California: And this is no wonder. A system failure at a financial firm or online retail store can cost millions of dollars in lost business.

Approximately 40 percent of computer outages are caused by operator errors. That is not because operators are not well-trained. It is because the technology is difficult to figure out, and IT managers are under pressure to make decisions in seconds.

So we are headed for a wall.

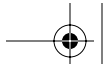
Businesses cannot roll-in processors and storage fast enough to avoid meltdowns when usage spikes, fend off viruses and hacker attacks, or manage the different operating systems that access information. People are good, but they are not that good.

Since no one is close to writing defect-free software or hardware, we need computers that are capable of running themselves, with far greater levels of intelligence built into the technology.

We are not talking about computers that can write the next "Ninth Symphony." We are talking about the same kind of intelligence we take for granted in our own bodies. We walk up three flights of stairs and our heart rate increases. It is hot, so we perspire. It is cold, so we shiver. We do not tell ourselves to do these things, they just happen.

If systems and networks adopt these attributes, managers could set business goals and computers would automatically set the IT actions needed to deliver them. For example, in a financial-trading environment, a manager might decide that trades have to be completed in less than a second to realize service and profitability goals. It would be up to software tools to configure the computer systems to meet those metrics. The implications for this "autonomic" business approach are immediately evident: A networking ser-





vice of organized, “smart” computing components that give us what we need, when we need it, without a conscious mental or even physical effort.

The goal is to increase the amount of automation that businesses need to leverage, extend, and sustain. Because the more that you can get human error out of the loop, the more efficient your business will become—whether you are a financial institution, a shipping company, an automotive manufacturer, an airline, or an online retailer. The beauty of it is that all of these complexities are hidden from the user.

The logic is compelling: Relief from the headaches of technology ownership and maintenance; an improved balance sheet; and, a much greater flexibility in meeting the demands of running a business.

However, in the end, perhaps the greatest benefit would be the freedom it would unlock. Sure, it will create enormous efficiencies. But the game-changing impact will be freeing up all companies—whether just starting out or well-established. The following figure shows the high-level problems being addressed and solved by Autonomic Computing.

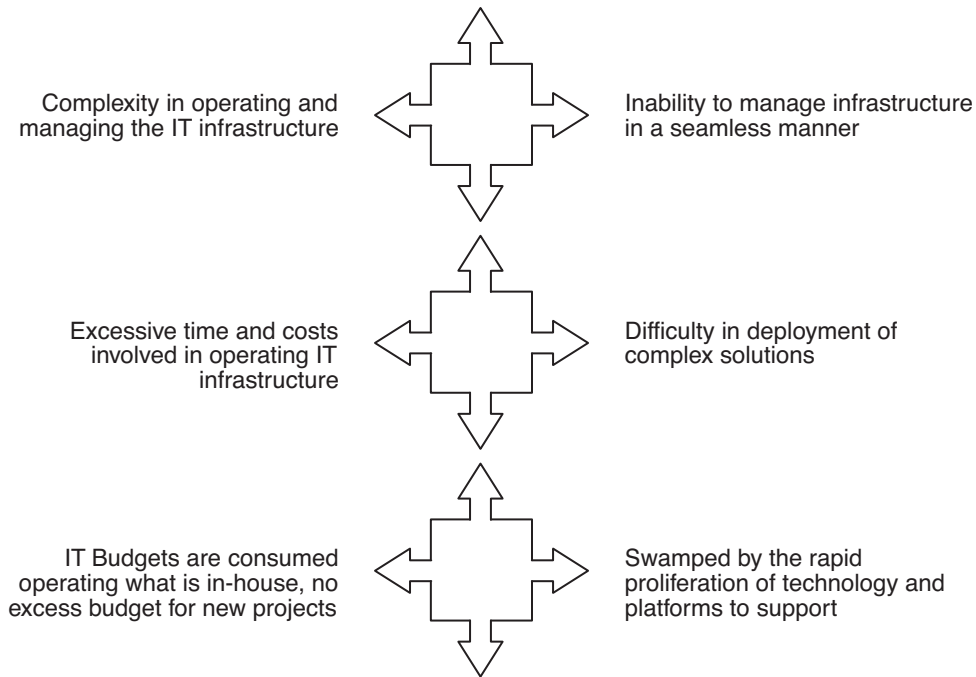
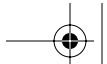


FIGURE 3.2 Where you start depends on the on demand business priorities. Increasing flexibility and reducing risk is the key—business models, processes, infrastructure, plus financing and delivery.





These are all outstanding insights that provide some of the strategic perspectives toward removing the mystery of Autonomic Computing.

THE AUTONOMIC GOAL IS SIMPLICITY

“The goal is to increase the amount of automation that businesses need to sustain. Because the more that you can get human error out of the loop, the more efficient your business will become—whether you are a financial institution, a shipping company, or an online retailer. The beauty of it is that all of these complexities are hidden from the user.”

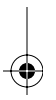
—Alan Ganek, Vice President of Autonomic Computing for
IBM Corporation’s Software Group.

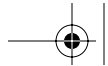
Perhaps you are now thinking about what would be involved for your company or organization to become an on demand business operation, with a multitude of Autonomic Computing transformation efforts successfully completed. Or, perhaps you are wondering what this would mean for the many employees in your company, should you execute such a transformation. If not, then you might be wondering how effectively you have been able to accomplish what the contents of this book address—establishing on demand business operations, including critical Autonomic Computing and maybe some key Grid Computing success stories.

Alan Ganek, in one of his public speaking engagements related to Autonomic Computing, mentions: “The goal of our Autonomic Computing initiative at IBM is to help you build more automated IT infrastructures to reduce costs, improve up-time, and make the most efficient use of increasingly scarce support skills to get new projects on line more rapidly.” This strategic perspective summarizes the fundamental values of Autonomic Computing.

To transform any organization or company to an on demand business operation, developers and IT professionals will be expected to help transform key areas of the technical infrastructure to support the integration of on demand processing. Business leaders will need to foster a way of thinking across the workforce to focus skills and critical thinking toward a common transformation goal. Managing innovation will become a staying thread for the marketing and sales teams, technology practitioners, and leading strategic thinkers. This is not an impossible task, nor is it a revolution, simply a transformation toward a more effective means of conducting business.

In the on demand business *flexible hosting* model, customers leverage on demand services only as they are required. The major resulting benefit from this flexible hosting-based approach is that the customer pays only for the “flexible hosting service” by the transaction, according to usage amounts, and is insulated from the core infrastructure used to deliver these types of





flexible hosting services. These flexible hosting approaches allows customers the ability to quickly and easily leverage many on demand business products and services to conduct their on demand business activities in a seamless manner.

Customer benefits realized in the on demand business flexible hosting model strategies include the ability to “pay as you go” for services. This improves the ability to better forecast technology needs, to be able to activate a service on demand at the point of need, and to access additional capacity and resources for only short periods of time. For example, a company could request autonomic capacity provisioning or server utilities if it suspects having to sustain increased levels of Web traffic (at any unknown time); later, this autonomic flexible hosting service capability could be turned off, as the increased Web traffic is realized and begins to reduce in intensity.

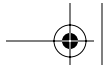
Although this flexible hosting approach to computing is very important and noteworthy to understand in the context of on demand business, this chapter will focus on the major fundamental concepts in the IBM on demand business strategy, Autonomic Computing, and less on IBM’s flexible hosting computing concepts.

Consider the fact that a realistic vision is an achievable mission. And, not surprisingly, many of today’s existing IT infrastructures are not staged for the kind of dynamic, responsive, integrated operating environment required for being a true on demand business enterprise. The IBM Corporation is, however, operating in an on demand Operating Environment today, and has carefully positioned the entire company to help others interested in achieving this same type of on demand Operating Environment. IBM continues to help many global customers develop a realistic vision, to create and execute on their own achievable on demand business mission.

At IBM, we recognize four essential transformation characteristics for the on demand Operating Environment. These characteristics are key for any company to consider, as it enters the on demand strategic transformation process. These on demand business transformation characteristics are defined as:

- *Integrated*—This is the key to the castle. Data must maintain its integrity and will require transaction processing of the highest order across custom applications throughout the enterprise. Instead of integrating “vertically” (within the operating system of the computer), applications will integrate “horizontally,” freeing them from the restrictive underpinnings of their underlying infrastructure.





- *Virtualized*—Companies will be able to access and pay for computing the same way they get electricity, telecommunications, or water—by the flip of a switch, the push of a button, the turn of a knob, or the click of a mouse. When traffic volumes or transactions unexpectedly spike, capacity can be added automatically over the 'Net. When things are quiet, your company pays less, and capital is then freed up to invest back into your business.
- *Open*—With most companies already having made huge investments in technology, the ability to “rip and replace” an entire system is just not an option. Open technical interfaces and agreed-upon standards are the only realistic way that many business processes, applications, and devices will be able to connect.
- *Autonomic*—The term “Autonomic Computing” is from the human anatomy’s autonomic nervous system. The same way we take for granted the human body’s management of breathing, digestion, and fending off viruses, companies will one day take for granted the network’s ability to manage, repair, and protect itself. Figure 3.3 shows control measures with resources.

On Demand Business Requires...

An IT operating environment that leverages open standards-based technologies, integrates across a heterogeneous environment, and capitalizes on autonomic capabilities

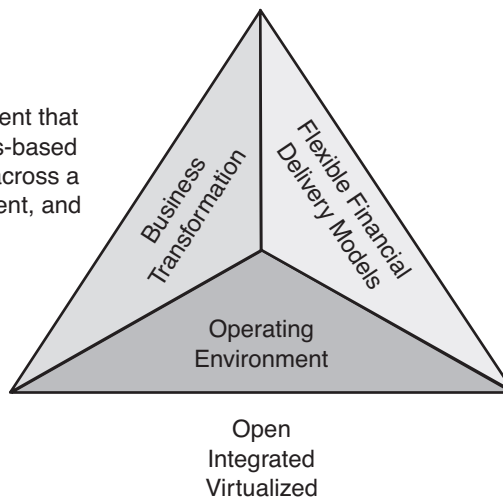
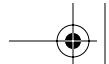


FIGURE 3.3 This illustration shows autonomic control measures as applied to resources (year 2000).

IBM has realized that *networking services* are the key enablers of all on demand business ecosystems, and these complex, autonomic networking services grids must be carefully considered in the evolutionary on demand business transformation. A key consideration here is the fact that networks





and the complex services they provide are critical to the success of any company in the on demand transformation. In on demand Operating Environments, especially concerning elements of Autonomic and Grid Computing, networking services quickly become a major dependency in the success of the overall mission, and are no longer a hidden assumption in the overall IT equation.

THE COSTS OF DOING BUSINESS

"A decade ago, hardware soaked up 80 percent of the cost of a data center," said Alan Ganek, Vice President, Autonomic Computing, IBM Software Group. *"Now, half the money spent goes for the people to run the center."*

More intuitive computers will provide a buffer for more complex IT systems, Ganek said.

The IBM Corporation's vision of Autonomic Computing embraces the development of intelligent, open systems and networks capable of running themselves, adapting to varying circumstances in accordance with business policies and objectives, and preparing resources to most efficiently handle the workloads placed on them. These Autonomic Computing systems manage complexity, "know" themselves, continuously tune themselves, and adapt to unpredictable conditions, all while seeking to prevent and recover from failures.

Just as the human body grows throughout life, the movement to a fully self-managing Autonomic Computing environment can only be realized on a gradual deployment model. IBM has been working for several years to make this long-range vision a reality with the introduction of advanced technologies in many current IBM products and services. These innovative developments in Autonomic Computing strategies and technologies deliver computing systems that offer both IBM and our global customers improved resiliency, accelerated implementation of new capabilities, and increased ROI in IT, while providing safe and secure operational environments.

ON DEMAND BUSINESS

To transform into an on demand business operation exploiting Autonomic Computing, one must manage transformation and effectively foster a new way of thinking. Senior technologists and key business leaders become the core of this evolutionary thinking process: exploration by many strategic thinkers of how Grid Computing and Autonomic Computing become fundamental in many aspects of the operational model. This leads to a fundamental operational graph, which everyone must be able to envision within his or her own domain(s).

A new "success" agenda for the entire workforce must be cascaded across the organization.

"Lead from the front" is the single most important message. This helps to build and substantiate the *on demand business roadmap* for success in your organization.



The *on demand business* evolution, incorporating Autonomic Computing, is not an overnight transformation in which system-wide, self-managing environments suddenly appear inside the infrastructure. Autonomic Computing must be a gradual transformation to deliver new technologies that are adopted and implemented at various stages and levels. These transformation levels are born at the most basic levels of the infrastructure and business, and transition across defined levels to fully autonomic operations.

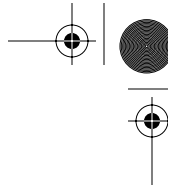
Autonomic Computing is focused on the most pressing problem facing the IT industry today: the increased complexity of an IT infrastructure that typically accompanies the increased business values delivered through computing advancements. This problem is contributing to an increasing inability of businesses to absorb new technology and solutions.

The scope of Autonomic Computing in any operating environment must address the complete domain of IT management complexities—marginal improvements will be afforded by individual technologies—or single operating environments will not be sufficient enough to sustain the computing advances that the IT user community requires and expects.

All elements of the IT system must be included. The deployment of business solutions invariably involves the coordination of all IT resources, including the following: Servers, storage, clients, middleware, applications, and network services. Furthermore, Autonomic Computing initiatives must insure growth in the levels of autonomic compliance in each of these areas. They must also be able to support all the platforms and/or vendors that supplied the IT infrastructure elements.

This multi-level autonomic journey will be an evolutionary process, while allowing each business to adopt Autonomic Computing capabilities in a non-obtrusive manner and at its own speed. Autonomic Computing delivers quantifiable savings and other qualitative values to customers as functions/features in the new version of products (both computing elements and management offerings). Multiplicative values will be realized when the Autonomic Computing capabilities of each product are integrated to deliver a fully autonomic system. Existing IT implementations will require a prescribed migration path/plan through which a fully Autonomic Computing environment and its associated benefits will be realized. The realization of an on demand Operating Environment must be driven from industry collaboration. The industry-wide advancement of Autonomic Computing technologies is based on open and de facto industry standards, which is the most feasible approach to transformation.





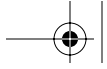
Paul Horn issued a challenge to the IT industry in 2001 to work together to solve the growing problem of complexity faced by IT customers. IBM has stepped up to that challenge and taken on the industry-wide leadership. This industry-wide leadership is what differentiates IBM from other suppliers' individual autonomic, adaptive, or organic computing initiatives. IBM's leadership makes sense, given the breadth and depth of IBM's experience across the most extensive product and services portfolio(s) in the industry, and IBM's open standards-based approach and commitment to work with its products and services across global heterogeneous environments.

IBM's mission, in addition to leading the industry in on demand business, is to establish IBM products as the best possible examples for Autonomic Computing that deliver distinct values and IBM services; and leads the facilitation for the adoption of Autonomic Computing technologies. This will be accomplished by dedicated focus and execution at an unprecedented level of integration across IBM, in response to Sam Palmisano's (IBM's Chairman and Chief Executive Officer) "*On Demand*" call to action in 2002.

The Autonomic Computing business strategy for supporting this transformation includes several very important Autonomic Computing initiatives. These perspectives are not only focused on IBM transformation accomplishments, but also are extensible to the entire IT industry, worldwide. These transformation initiatives are as follows:

1. An overarching architectural framework, or blueprint, and corresponding open standards that underpin the industry's development of autonomic technologies
2. An IT deployment model that defines each progressive level of autonomic maturity
3. New, integrating core Autonomic Computing technologies that when combined with existing products, serve to fulfill the architecture and deliver on the vision of the on demand Operating Environment
4. Autonomic Computing technological enhancements to existing product lines that conform to the architecture and standards that enable level progression
5. Autonomic Computing service offerings that can define the roadmap of required on demand business initiatives for any given business and deliver the technology and services as prescribed by those initiatives
6. Customer/partner programs to co-create standards and technologies and validate the openness of our initiative to get the most relevant and complete self-managing systems to market sooner





The following points explore each of these Autonomic Computing perspectives, which are extensible across all global industries. Consider deliverables such as the following in your organization:

1. *An overarching architectural framework, or blueprint, and corresponding open standards that underpin the industry's development of autonomic technologies*
Architecture, technology, and standards deliverables drive the industry toward a common technical vision of Autonomic Computing. This *Autonomic Computing Blueprint* will be an overarching view of the technology underpinning Autonomic Computing. These technological underpinnings depict how the various key technologies, interfaces, services, functions, and capabilities all fit together to form an end-to-end technical framework. The purpose of the framework is to set the context of what is meant, technically, by the terms of Autonomic Computing, and to show relationships. This framework is an active enabler in the quest to realize the on demand business vision. This is discussed in more detail in later sections of this chapter.
2. *An IT deployment model that defines each progressive level of autonomic maturity*

As we introduce the need for business transformation in these early chapters, let us again review this evolution. The following list and Figure 3.4 prescribe these five levels of transformation toward achieving a refined state of Autonomic Computing, which is required in every on demand Operating Environment. These levels are as follows:

- *Level 1: Basic*—The starting point where most systems are today, this level represents manual computing, in which all system elements are managed independently by an extensive, highly skilled IT staff. The staff sets up, monitors, and eventually replaces system elements.
- *Level 2: Managed*—Systems management technologies can be used to collect and consolidate information from disparate systems onto fewer consoles, reducing administrative time. There is greater system awareness and improved productivity.
- *Level 3: Predictive*—The system monitors and correlates data to recognize patterns and recommends actions that are approved and initiated by the IT staff. This reduces the dependency on deep skills and enables faster and better decision-making.
- *Level 4: Adaptive*—In addition to monitoring and correlating data, the system takes action based on the information. This can be mapped to SLAs, thereby enhancing IT agility and resiliency with minimal human interaction while insuring that the SLAs are met.



- *Level 5: Autonomic*—Fully integrated systems and components are dynamically managed by business rules and policies, enabling IT staff to focus on meeting business needs with true business agility and resiliency.

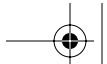
	Basic Level 1	Managed Level 2	Predictive Level 3	Adaptive Level 4	Autonomic Level 5
Characteristics	Multiple sources of system generated data	Consolidation of data and actions through management tools	Systems monitors, correlates and recommends actions	System monitors, correlates and takes action	Integrated components dynamically managed by business rules/policies
Skills	Requires extensive, highly skilled IT staff	IT staff analyzes and takes actions	IT staff approves and initiates actions	IT staff manages performance against SLAs	IT staff focuses on enabling business needs
Benefits	Basic requirements Met	Greater system awareness Improved productivity	Reduced dependency on deep skills Faster/better decision making	Balanced human/system interaction IT agility and resiliency	Business policy drives IT management Business agility and resiliency
Manual					Autonomic

FIGURE 3.4 The various transformation levels of Autonomic Computing.

Although we will go into more detail on these autonomic levels of transformation in the forthcoming section titled “An Architecture Blueprint for Autonomic Computing,” let us begin to explore these levels now to better understand operational positioning. Consider the following as a map, or route, for the future of driving efficiencies into business enterprise operations. To assist one in this assessment, IBM is also publishing guidelines and “adoption models.” In summary, these five levels show this evolution (not revolution) based on one’s need for investment protection.

Delivering autonomic IT infrastructures is, indeed, an evolutionary process that is enabled by technology; however, this transformation is ultimately implemented by each enterprise through the adoption of these technologies, along with supporting business processes and the proper critical skills.

IBM views these five levels as a map of the future state of business for any enterprise engaged in the on demand business journey. The repre-



sentation of these levels starts at the basic level and progresses through the managed, predictive, and adaptive levels, and finally to the fully autonomic level.

The first level, the *basic level*, represents the starting point where many IT infrastructures are today. At this level, one will note that IT professionals who set it up, monitor it, and eventually replace it will typically manage each element in the infrastructure independently.

At the second level, the *managed level*, systems management technologies can be utilized to collect information from disparate systems onto fewer consoles, thus reducing the time it takes for the administrator to collect and synthesize information as the systems become more complex to operate.

At the third level, *predictive*, as new technologies are introduced that provide correlation among several elements, the infrastructure itself can begin to recognize patterns, predict the optimal configuration, and provide advice on what course of action the administrator should execute.

As these technologies improve and as people become more comfortable with the advice and predictive power of this infrastructure, we can progress to the fourth level, the *adaptive level*. At this level, the elements themselves, via *closed loop automation*, can automatically take the right course of action based on the information available to them, and the knowledge and state of what is actually occurring in the infrastructure.

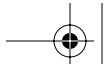
Finally, in this five-level transformation process, to achieve the fully *autonomic level*, the fifth level, the IT infrastructure must be governed by business policies and objectives.

Many industry analysts deem this multi-level “evolution” as the right approach to Autonomic Computing, ultimately delivering the arrival of companies to fully enabled on demand Operating Environments.

3. *New, integrating core Autonomic Computing technologies that when combined with existing products, serve to fulfill the architecture and deliver on the vision of the on demand Operating Environment*

Integrating core Autonomic Computing technologies will serve as the fundamental common “building blocks,” ensuring consistent implementations and behaviors of autonomic systems. This will also facilitate the integration and interoperability of many heterogeneous components. After surveying what autonomic functionalities exist in the marketplace, a pattern to meet one’s business requirements will become obvious across most technologies. This pattern will be related to individual products, services, or operating environments.





To realize the autonomic vision in the majority of customer IT implementations that are heterogeneous in nature (i.e., multiple products across multiple operating systems), the strategy must include architectures and standards for the creation of the integrating core Autonomic Computing technologies; for example, technologies allowing end-to-end self-management capabilities across the heterogeneous environment. Furthermore, the very business processes that necessitated the need for the heterogeneous mix of technology to begin with must now ultimately drive these management actions. The evolution of technology also requires the evolution of processes, skills, and their respective linkages to achieve the on demand Operating Environment. Figure 3.5 shows a management interface.

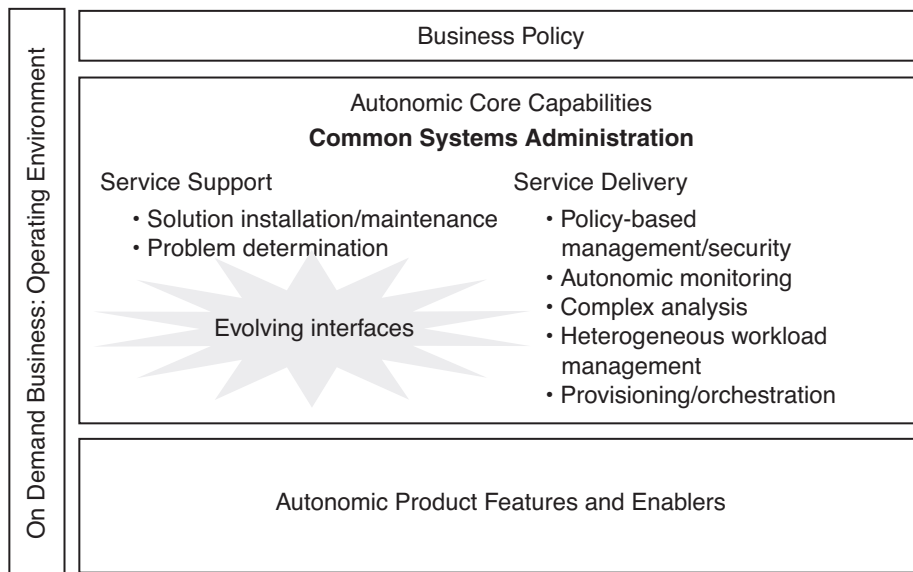
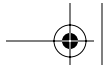


FIGURE 3.5 Core capabilities for enabling Autonomic Computing.

The following examples of Autonomic Computing core technologies illustrate why they are so critical as key foundational elements of the overall strategic approach.

The first example is a set of integrating core technologies required in the area of problem determination. The industry today is largely represented by a vast number of individual, product-unique approaches to problem identification and resolution. Even within individual products, problem determination is often inconsistent and incomplete in its approach and implementation. In most cases, a product’s problem determination con-





structs have the appearance of an afterthought, as opposed to the product of a well-thought-out, robust, and extensible design.

This “afterthought” problem is a systemic and pervasive concern across many industrial solutions. So often, the focus is on advancing, new, and leading-edge technology or functionality. And then, in retrospect, the field support of that technology remains as an afterthought, perhaps based on the assumption that the technology will be developed without fault.

Consider for a minute some comments made by a college student at one of the top engineering universities in the U.S. This discussion concerns the student’s participation in a robotics project as part of an international competition. The robot was able to perform a number of very impressive maneuvers, employing some truly innovative ideas that were implemented by the development team. As we discussed the project, there was a lot of excitement and passion for the newly created abilities of the robot, but far less enthusiasm when the conversation turned toward robot maintenance.

The need, therefore, is to define, standardize, and integrate an industry-wide approach to problem determination to achieve *self-healing* in multi-component environments. This will always consist of a standards-based approach to data capture, analysis, and remediation to realize the self-healing aspects, and would practically be achieved by a phased approach, over time, represented by incremental levels of increasing autonomic maturity.

The first step is to get the right data from the system, in a consistent, standards-based format. The next step consists of putting a set of symptoms and corresponding actionable causes in a consistent format, and building tools that can correlate the data to match against a cross-product, standards-based problem/symptom database: in other words, autonomic event correlation.

We ultimately want to automate fixing defects by being able to automate the provisioning of an application with standardized fixes (or temporary workarounds) based on the business policies that govern each application.

While we evolve IT infrastructures toward self-healing, there are many benefits that can be realized and are already having an impact. Here are two examples: First, the common format for log entries, submitted as a standard, is dramatically reducing training time for administrators and providing a consistent format to evaluate multiple logs together. Second, automated correlation engines are reducing manual analysis by providing a programmatic method to correlate the logs that are



adapted to the common format. Figure 3.6 shows autonomic nervous system compared to the autonomic model.

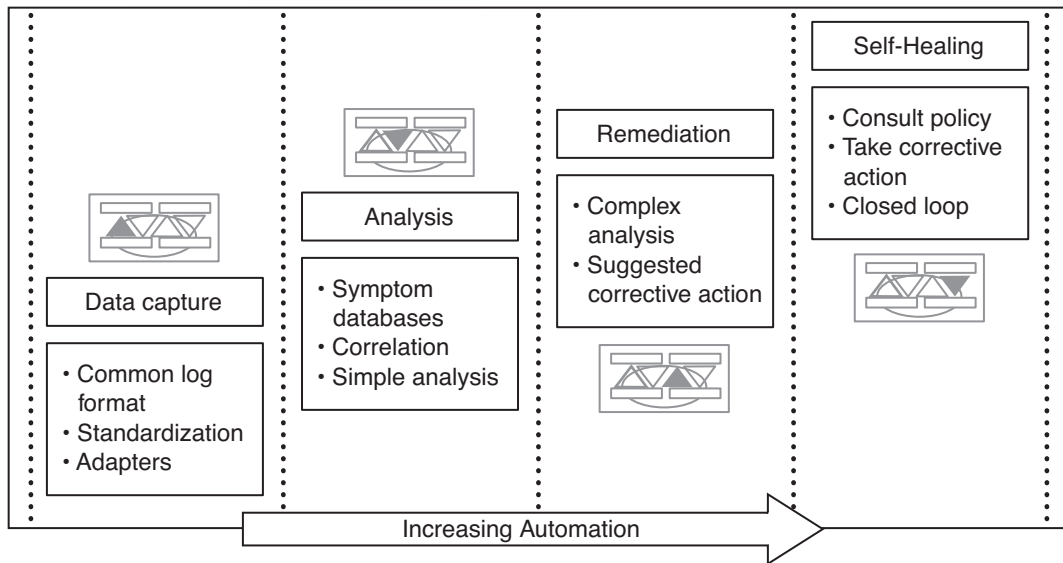
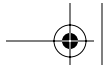


FIGURE 3.6 The various levels involved in creating self-healing systems.

Other examples, which are discussed in more detail later in this chapter, include:

- An integrated solutions console for common system administration that addresses the complexity of operating multiple heterogeneous products, each with its own end-user interface. The idea here is to provide one consistent administration user interface for use across an IT infrastructure product portfolio using WebSphere Portal Server as a basis. This would include the provision of a common runtime infrastructure and development tools based on industry standards.
- Consistent software installation technology across all products that provides solution packaging standards for defining complete installation end-to-end solutions and consistent and up-to-date configuration and dependency data, which are key to building self-configuring autonomic systems.
- Policy tools for policy-based management that provide uniform cross-product policy definition and management infrastructure needed for delivering system-wide self-management capabilities that map automated actions to the needs of the business. This would include the



development of the ability to describe business needs, rules, and policies in system-readable form.

- Workload balancing across heterogeneous systems requires infrastructure enablement, allowing business transactions to operate in a resource-balanced way across server pools (e.g., in heterogeneous environments).
- Policy-based, intelligent orchestration and provisioning that drive infrastructure provisioning, capacity management, and service-level delivery across the automation environment and enable the ability to rapidly deploy a complete application environment. This allows one to re-purpose computing resources for uses such as application staging, load testing, and support of anticipated peaks in application demand.

Integrating such core technologies will have a profound effect on the reduction of complexities (and costs) for the customer, and also for the supplier, which can take advantage of component reuse as well as having a dramatic impact on service costs. These cost savings can then be applied to continue to fund the advancement of autonomic technologies and their respective implementations.

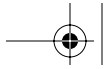
The successful adoption of these technologies will also require corresponding adjustments to the IT processes and skills that are impacted by these core technologies. This must be a conscious effort that begins at the early design stage of both the development of the technologies as well as the customer adoption of these technologies. The creation of flow sequence diagrams is required to document the interaction patterns between humans and products in the accomplishment of service and system management. These sequence diagrams will be used to identify requirements for IBM and industry products to improve the value of Autonomic Computing core technologies and Autonomic Computing functions and features.

4. *Autonomic Computing technological enhancements to existing product lines that conform to the architecture and standards that enable level progression*

The work must take place across IBM and the industry to deliver industry-leading Autonomic Computing functions and features while simultaneously insuring existing product offerings can be easily integrated within an autonomic system environment. This includes driving continued implementation of specific self-management product capabilities into product offerings along four dimensions, as depicted in Figure 3.7:

- *Self-configuration* features to provide greater ease of use and increased availability





- *Self-healing* functions to prevent customer downtime
- *Self-optimizing* functions to provide the highest utilization of customer resources
- *Self-protecting* functions to safeguard customer access to system resources and protect customer data

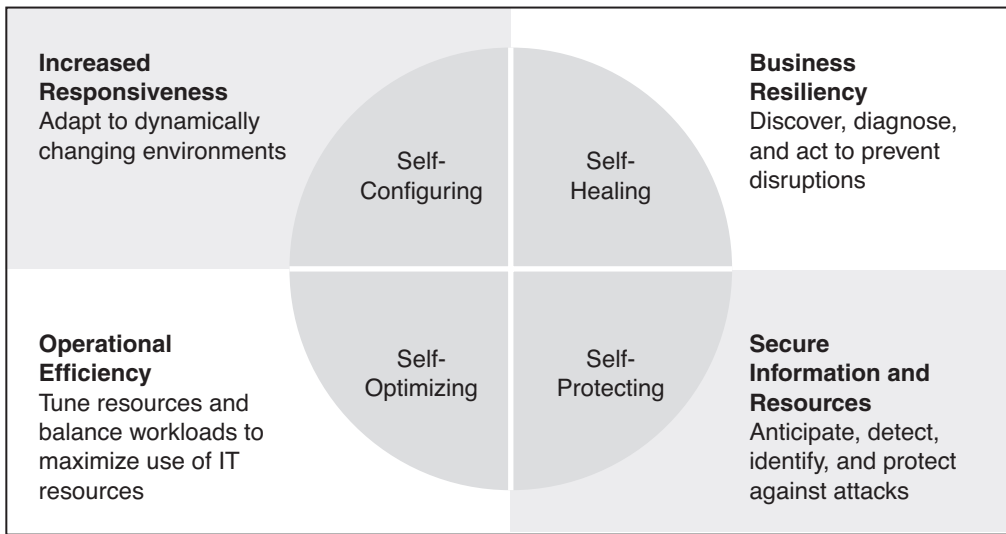


FIGURE 3.7 The various Autonomic Computing self-managing capabilities.

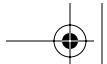
Additionally, it is important to enable the integration of product elements within an autonomic system environment by ensuring adoption of common standards. This is accomplished by applying autonomic core technologies and compliance with strategic Autonomic Computing architecture guidelines for evolving autonomic system behaviors.

Here are some examples of how Autonomic Computing technologies enable IBM brand offerings:

Servers—Servers that respond to unexpected capacity demands and system errors, without human intervention, yielding dramatic improvements in the server’s reliability, availability, and serviceability while simultaneously significantly reducing both downtime and cost of ownership.

Storage—Storage systems that utilize hot spot detection and online data migration techniques to guide object placement (e.g., data and applications) in a shared storage pool. The details of low-level object-to-device assignment, disk space provisioning, automated failure detection and



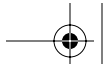


recovery, and storage network traffic management are handled invisibly by the system while providing an accurate depiction of storage capacity and bandwidth trends.

Software—Self-diagnosing, self-repairing, and self-managing behaviors in software that will yield more cost-effective customer service and prevent unethical hacking attacks, as well as IT systems that operate together more efficiently in the accomplishment of business objectives. For example:

- *Integration middleware software*—The system will allow users to express policy and what they believe should happen, and then the software will drive the execution. Pre-emptive diagnostics will automatically recognize and solve problems such as configuration settings, software updates, provisioning, and load balancing across application server clusters, and intercept/block unauthorized system calls.
- *Data management software*—Sophisticated database management tools such as performance and health monitors, recovery experts, and configuration advisors are complemented by capabilities that result in operating parameters changing in real time to better address changes in the environment or business priorities.
- *Systems management software*—Automatically deploy, update, track, repair, and manage equipment, software, and configuration changes to provide better alignment between business priorities and IT.











Systems Management		<ul style="list-style-type: none"> • Access/Identify/Risk Managers • Storage Resource Manager • Service Level Advisor
Client		<ul style="list-style-type: none"> • ImageUltra • Rapid Restore PC • Embedded Security Subsystem
Application		<ul style="list-style-type: none"> • Prioritization of User Transactions • EdgeServer Performance Optimization • Problem Analysis and Recovery
Database & Collaboration		<ul style="list-style-type: none"> • DB2 Query Patroller • DB2 Configuration Advisor • Tivoli Analyzer for Domino
Servers		<ul style="list-style-type: none"> • Dynamic Partitioning • IBM Director • BladeCenter • Electronic Service Agent
Storage		<ul style="list-style-type: none"> • Intelligent Cache Configuration • Predictable Failure Analysis • Dynamic Volume Expansion

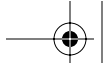
FIGURE 3.8 IBM Autonomic product capability examples.

It is helpful to employ metrics to guide and track progress in moving products forward toward Autonomic Computing goals. This will include tracking new autonomic features, adoption of core technologies, and measurements that reflect the reduced complexity achieved by each release in deploying and using the products.

5. *Autonomic Computing service offerings that can define the roadmap of required on demand business initiatives for any given business and deliver the technology and services as prescribed by those initiatives*

Automation methodologies and tooling for systems integrators communicate the benefits of using new autonomic technologies as they become available in the delivery of customer solutions, providing the enterprise and its partners with Autonomic Computing service offerings. This will help solve specific customer problems as these new technologies become available, while providing insights as to how these technologies can best be utilized in various scenarios.





To this end, IBM has developed an Autonomic Computing Adoption Model assessment, which is a six- to eight-week holistic assessment of the autonomic maturity of an IT implementation. This provides the most practical set of initiatives required to advance the state of the on demand Operating Environment. The assessment encompasses not only the technology, but also the processes that surround the technology and the required adjustment of skills and organizational constructs. This Adoption Model is based on experiences from IBM customer engagements, and leverages best practices for managing systems while providing the opportunity to leverage the most recent Autonomic Computing core technologies from IBM and industry partners.

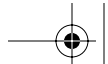
6. Customer/partner programs to co-create standards and technologies and validate the openness of initiatives to get the most relevant and complete self-managing systems to market sooner

Partnerships through customer and business partner programs provide the opportunity to get more complete and relevant self-managing systems to market sooner. The partner ecosystem of Autonomic Computing runs the gamut, from element vendors (e.g., servers, storage, middleware, and network services and equipment) to systems integrators, to system management partners, development information systems vendors (ISVs), and channel partners. IBM has well-established partner programs that are being leveraged to engage partners through joint collaboration, thus actively engaging the industry as a whole through participation in standards-based work.

An obvious element of the IBM strategic perspectives is to build on the industry-wide on demand business vision to rally participation and endorsement from other enterprises in global industries. Autonomic Computing covers a broad spectrum of partners, including:

- Vendors that provide technology at all levels of the customer solution stack on concepts, architecture, and standards, and how to best leverage each product and solution.
- Systems integrators that provide services and implement network designs, including hardware and software, which incorporate Autonomic Computing specifications and provide technical support post-installation.
- Development ISVs that license Autonomic Computing technology for embedding within broader customer solutions. (In some cases, IBM's technology is directly embedded; however, in other cases, joint development and integration are needed to accelerate time to market.)



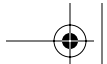


The adoption rate of the Autonomic Computing architecture and related core technologies depends on the IT industry's software developers having relevant open standards. It also depends on open access to architectures and integrating technologies in the form of toolkits, or Software Development Kits (SDKs), and partner programs that make it easy to incorporate them into their products. Partner enablement programs have been established using IBM Solution Partnership Centers to enable ISVs to code, port, and test their relevant Autonomic Computing functions/features for use in autonomic solution offerings. IBM also provides individual technical assistance for partners, and wherever possible, develops references and joint marketing materials that promote the concepts of Autonomic Computing and on demand business.

While engaged in the provisioning of partner enablement technologies, it is important to supply SDKs to make it easier for industry partners operating in the Autonomic Computing arena to align with Autonomic Computing interfaces and information architectures. Autonomic Computing toolkits are utilized to build autonomic managers that implement Autonomic Computing specifications; allowing vendors to easily adapt their products according to the specifications also utilize them. IBM on demand business design centers have also proven to be highly effective in communicating Autonomic Computing messages to prospective customers, and engaging with customers in design workshops, solution assessments, and developing "proofs of concept" for their Autonomic Computing applications. This includes technologies and architectures in the solution set of any customer-defined problem. In addition to individual or group-facilitated meetings, continuous education, distance learning, and collaboration of the Autonomic Computing business partner community are facilitated via events such as PartnerWorld and developerWorks.

It is important to IBM and other enterprises to facilitate the creation of customer and partner references that serve as "lighthouses" to demonstrate what is possible to achieve with Autonomic Computing and on demand business. These activities serve as catalysts for further advancements. It is important within IBM to drive early adoption of Autonomic Computing and Grid Computing-based technologies and solutions. IBM does this by virtue of the delivery of Autonomic Computing and Grid Computing services/solutions to IBM employees to demonstrate commitment and leadership by example. IBM is (and has been for some time) currently engaged in extensive projects to do just that.





The good news is that many of the inventions required to build autonomic technologies are available and need only to be formalized and assimilated through the mechanism of standards-based collaboration across the industry. For those technologies that have yet to be developed, IBM is making significant investments in research, joint university programs, and industry collaboration projects. IBM will drive the rapid development of on demand business through such leadership initiatives; the Autonomic Computing Organization, a highly focused, cross-corporate business unit within IBM, is one organization deeply involved in these types of activities.

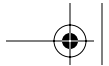
Another facilitator in the industry is Web-accessible content through various existing IBM "e-venues," such as developerWorks and alphaWorks, as well as via IBM's Autonomic Computing Web site, which can be found at www.ibm.com/autonomic.

The on demand business is focused on realizing efficiencies in business operations and improving ROI. This is true whether one is the provider of on demand business products and services or the recipient of on demand business products and services. In the early 1990s, the Internet linked scientists across academia, government, and research. The Internet then evolved to provide e-mail, and then the World Wide Web, which was excellent for communicating marketing messages, but did not have the demonstrative capabilities of the values we are noting today with on demand Operating Environments.

IT has matured, enabling a plurality of global electronic commerce solutions and advanced Web services across the Internet. This is presented in thought-provoking detail in the book *Exploring e-Commerce: Global e-Business and e-Societies* [Fellenstein01]. In the late 1990s, with e-Commerce still driving new business processes and seemingly brilliant new start-up companies, the world encountered a tremendous stock market boom. However, this new horizon and capability came with a cost; it has experienced some serious investments in technology. The on demand business has since evolved and has now been instantiated with new advanced forms of Web services. Conducting on demand business is a new technological and business operations frontier, yielding advanced new capabilities and services provider models, which help to strengthen service level delivery capabilities and QoS.

When any business invests in IT, it clearly expects to derive credible benefits from its investments. Issues surrounding the needs for ongoing cost reductions will never disappear, nor will issues surrounding technology advancements, integration, and automation. To become an on demand business is not a revolution, nor a short-term strategic endeavor; it is, simply stated, a





very precise strategic *evolution*. Implementing on demand business practices, including Autonomic Computing, is not a short-lived business trend; these advanced topics represent the combined manifestation of incredibly powerful business leadership and carefully planned strategic investments. This notion of becoming an on demand Operating Environment is a new way of operating that embodies transformed business processes, utilizing advanced services, while implementing very effective operational environment capabilities.

The Autonomic Computing Vision

The following discussion surrounding the Autonomic Computing vision explores a broad set of ideas and approaches to Autonomic Computing, as well as some first steps in what we at IBM see as a journey for our customers to create more self-managing computing system environments.

Autonomic Computing represents a collection and integration of technologies and business processes, which in turn enhances the creation of an IT computing infrastructure in many beneficial ways.

ONE MAJOR GOAL OF AUTONOMIC COMPUTING

A fundamental goal of Autonomic Computing is to provide *self-configuration* capabilities for the entire IT infrastructure, not just individual servers, software, and storage devices.

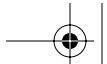
This conceptual goal extends far beyond traditional approaches found in many of today's steady-state environments.

As published in the *IBM Systems Journal* dedicated to Autonomic Computing [IBMSYSJ], the article entitled "The dawning of the Autonomic Computing era," [Ganek02] very eloquently describes the new, innovative discipline of Autonomic Computing. In this section, we will provide the reprint of this innovative work, which outlines some key Autonomic Computing perspectives.

In the following discussion, we will explore this new and innovative era of computing with an overview of IBM's Autonomic Computing efforts. These concepts will explore some of IBM's strategic thoughts surrounding the subject of on demand business. Many of the key strategic thinkers in this area, such as Ric Telford, John Sweitzer, and Jim Crosskey of IBM, have contributed in very significant ways to these On Demand perspectives.

This article will explore the industry and marketplace drivers, the fundamental characteristics of autonomic systems, a framework for how systems will evolve to achieve a more self-managing state, and the key roles of the





open industry standards necessary to support autonomic behavior in heterogeneous system environments.

THE NEXT SECTION IS AN EARLY REPRINT OF [GANEK02].

This *IBM Systems Journal* article, “*The dawning of the Autonomic Computing era*,” exactly as published by Alan Ganek and Thomas A. Corbi, sets a precedent for one of the two key underpinnings of On Demand computing. This particular strategic underpinning, Autonomic Computing, will be further referenced throughout this book.

IBM Systems Journal, Vol. 42, No. 1. pp. 5-18. Copyright © 2003 International Business Machines Corporation. Reprinted with permission from *IBM Systems Journal*, Vol. 42, No. 1.

THE DAWNING OF THE AUTONOMIC COMPUTING ERA

On March 8, 2001, Paul Horn, IBM Senior Vice President and Director of Research, presented the theme and importance of Autonomic Computing to the National Academy of Engineering at Harvard University. ⁽¹⁾

One month later, Irving Wladawsky-Berger, Vice President of Strategy and Technology for the IBM Server Group, introduced the Server Group’s Autonomic Computing project (then named eLiza*)⁽²⁾ with the goal of providing self-managing systems to address those concerns. Thus began IBM’s commitment to deliver “Autonomic Computing”—a new company-wide and, it is to be hoped, industry-wide, initiative targeted at coping with the rapidly growing complexity of operating, managing, and integrating computing systems.

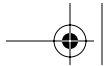
INFORMATION TECHNOLOGY AND THE IMPOSSIBLE PROBLEM

“The information technology industry loves to prove the impossible possible. We obliterate barriers and set records with astonishing regularity. But now we face a problem springing from the very core of our success—and too few of us are focused on solving it. More than any other I/T problem, this one—if it remains unsolved—will actually prevent us from moving to the next era of computing. The obstacle is complexity. Dealing with it is the single most important challenge facing the I/T industry.”

—Paul Horn, IBM Senior Vice President and Director of Research,
presented this theme and the importance of
Autonomic Computing to the
National Academy of Engineering at Harvard University on March 8, 2001.

We do not see a change in Moore’s law ⁽³⁾ that would slow development as the main obstacle to further progress in the information technology (IT) industry. Rather, it is the IT industry’s exploitation of the technologies in accordance with Moore’s law that has led to the verge of a complexity crisis. Software developers have fully exploited a four- to six-orders-of-magnitude increase in computational power—producing ever more sophisticated soft-





ware applications and environments. There has been exponential growth in the number and variety of systems and components. The value of database technology and the Internet has fueled significant growth in storage subsystems to hold petabytes⁽⁴⁾ of structured and unstructured information. Networks have interconnected the distributed, heterogeneous systems of the IT industry. Our information society creates unpredictable and highly variable workloads on those networked systems. And today, those increasingly valuable, complex systems require more and more skilled IT professionals to install, configure, operate, tune, and maintain them.

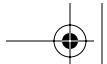
IBM is using the phrase “Autonomic Computing”⁽⁵⁾ to represent the vision of how IBM, the rest of the IT industry, academia, and the national laboratories can address this new challenge. By choosing the word “autonomic,” IBM is making an analogy with the autonomic nervous system. The autonomic nervous system frees our conscious brain from the burden of having to deal with vital but lower-level functions. Autonomic computing will free system administrators from many of today’s routine management and operational tasks. Corporations will be able to devote more of their IT skills toward fulfilling the needs of their core businesses, instead of having to spend an increasing amount of time dealing with the complexity of computing systems.

Need for Autonomic Computing As Frederick P. Brooks, Jr., one of the architects of the IBM System/360*, observed, “Complexity is the business we are in, and complexity is what limits us.”⁽⁶⁾ The computer industry has spent decades creating systems of marvelous and ever-increasing complexity. But today, complexity itself is the problem.

The spiraling cost of managing the increasing complexity of computing systems is becoming a significant inhibitor that threatens to undermine the future growth and societal benefits of information technology. Simply stated, managing complex systems has grown too costly and prone to error. Administering a myriad of system management details is too labor-intensive. People under such pressure make mistakes, increasing the potential of system outages with a concurrent impact on business. And, testing and tuning complex systems is becoming more difficult. Consider:

- It is now estimated that one-third to one-half of a company’s total IT budget is spent preventing or recovering from crashes.⁽⁷⁾ Nick Tabetlion, CTO of Fujitsu Softek, said: “The commonly used number is: For every dollar to purchase storage, you spend \$9 to have someone manage it.”⁽⁸⁾





- Aberdeen Group studies show that administrative cost can account for 60 to 75 percent of the overall cost of database ownership (this includes administrative tools, installation, upgrade and deployment, training, administrator salaries, and service and support from database suppliers).⁽⁹⁾
- When you examine data on the root cause of computer system outages, you find that about 40 percent are caused by operator error,⁽¹⁰⁾ and the reason is not because operators are not well-trained or do not have the right capabilities. Rather, it is because the complexities of today's computer systems are too difficult to understand, and IT operators and managers are under pressure to make decisions about problems in seconds.⁽¹¹⁾
- A Yankee Group report⁽¹²⁾ estimated that downtime caused by security incidents cost as much as \$4,500,000 per hour for brokerages and \$2,600,000 for banking firms.
- David J. Clancy, chief of the Computational Sciences Division at the NASA Ames Research Center, underscored the problem of the increasing systems complexity issues: "Forty percent of the group's software work is devoted to test," he said, and added, "As the range of behavior of a system grows, the test problem grows exponentially."⁽¹³⁾
- A recent Meta Group study looked at the impact of downtime by industry sector as shown in Figure 1.

Although estimated, cost data such as shown in Figure 1 are indicative of the economic impact of system failures and downtime. According to a recent IT resource survey by the Merit Project of Computer Associates International, 1867 respondents grouped the most common causes of outages into four areas of data center operations: systems, networks, database, and applications.⁽¹⁴⁾ Most frequently cited outages included:

- For systems: operational error, user error, third-party software error, internally developed software problem, inadequate change control, lack of automated processes
- For networks: performance overload, peak load problems, insufficient bandwidth
- For database: out of disk space, log file full, performance overload
- For applications: application error, inadequate change control, operational error, non-automated application exceptions



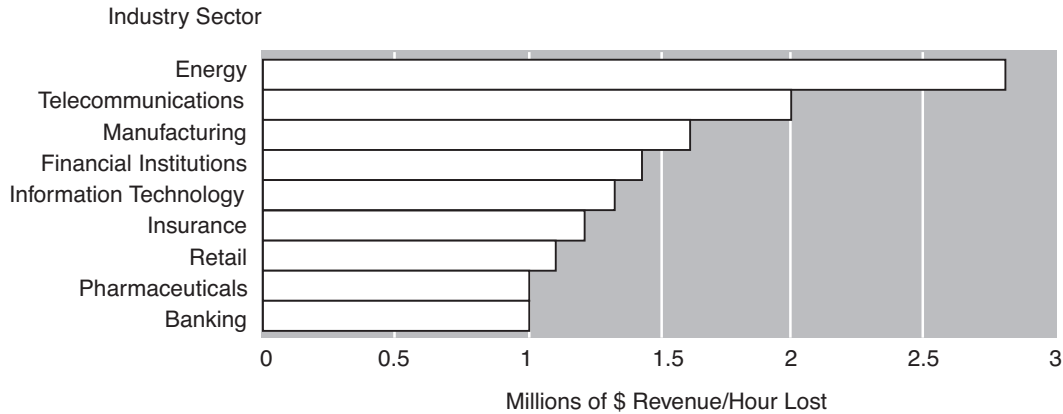
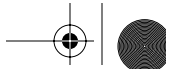


Figure 1 The illustration depicts data from *IT Performance Engineering and Measurement Strategies: Quantifying Performance Loss*, Meta Group, Stamford, Connecticut, USA (October 2000).

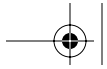
Well-engineered autonomic functions targeted at improving and automating systems operations, installation, dependency management, and performance management can address many causes of these “most frequent” outages and reduce outages and downtime.

Confluences of marketplace forces are driving the industry toward Autonomic Computing. Complex heterogeneous infrastructures composed of dozens of applications, hundreds of system components, and thousands of tuning parameters are a reality. New business models depend on the IT infrastructure being available 24 hours a day, 7 days a week. In the face of an economic downturn, there is an increasing management focus on “return on investment” and operational cost controls—while staffing costs exceed the costs of technology. To compound matters further, there continues to be a scarcity of highly skilled IT professionals to install, configure, optimize, and maintain these complex, heterogeneous systems.

To respond, system design objectives must shift from the “pure” price/performance requirements to issues of robustness and manageability in the total-cost-of-ownership equation. As a profession, we must strive to simplify and automate the management of systems. Today’s systems must evolve to become much more self-managing, that is: self-configuring, self-healing, self-optimizing, and self-protecting.

Dr. Irving Wladawsky-Berger outlined the solution at the Kennedy Consulting Summit in November 2001: “There is only one answer: The technology needs to manage itself. Now, we do not mean any far out AI {*Artificial Intelligence*} project; what we mean is that we need to develop the right software,





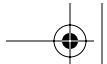
the right architecture, the right mechanisms . . . So that instead of the technology behaving in its usual pedantic way and requiring a human being to do everything for it, it starts behaving more like the ‘intelligent’ computer we all expect it to be, and starts taking care of its own needs. If it does not feel well, it does something. If someone is attacking it, the system recognizes it and deals with the attack. If it needs more computing power, it just goes and gets it, and it does not keep looking for human beings to step in.”⁽¹⁵⁾

What is Autonomic Computing? Automating the management of computing resources is not a new problem for computer scientists. For decades system components and software have been evolving to deal with the increased complexity of system control, resource sharing, and operational management. Autonomic computing is just the next logical evolution of these past trends to address the increasingly complex and distributed computing environments of today. So why then is this something new? Why a call to arms to the industry for heightened focus and new approaches? The answer lies in the radical changes in the information technology environment in just the few short years since the mid-1990s, with the use of the Internet and Business extending environments to a dramatically larger scale, broader reach, and a more mission-critical fundamental requirement for business. In that time the norm for a large on-line system has escalated from applications such as networks consisting of tens of thousands of fixed-function automated teller machines connected over private networks to rich suites of financial services applications that can be accessed via a wide range of devices (personal computer, notebook, handheld device, smart phone, smart card, etc.) by tens of millions of people worldwide over the Internet.

IBM’s Autonomic Computing initiative has been outlined broadly. Paul Horn⁽¹⁾ described this “grand challenge” and called for industry-wide collaboration toward developing Autonomic Computing systems that have characteristics as follows:

- To be autonomic, a system needs to “know itself”—and consist of components that also possess a system identity.
- An autonomic system must configure and reconfigure itself under varying and unpredictable conditions.
- An autonomic system never settles for the status quo—it always looks for ways to optimize its workings.
- An autonomic system must perform something akin to healing—it must be able to recover from routine and extraordinary events that might cause some parts to malfunction.





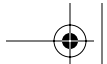
- A virtual world is no less dangerous than the physical one, so an Autonomic Computing system must be an expert in self-protection.
- An Autonomic Computing system knows its environment and the context surrounding its activity, and acts accordingly.
- An autonomic system cannot exist in a hermetic environment (and must adhere to open standards).
- Perhaps most critical for the user, an Autonomic Computing system will anticipate the optimized resources needed to meet a user's information needs while keeping its complexity hidden.

Fundamentals of Autonomic Computing. In order to incorporate these characteristics in “self-managing” systems, future Autonomic Computing systems will have four fundamental features. Various aspects of these four fundamental “self” properties are further explored in the “Autonomic Computing” *IBM Systems Journal*, Volume 42, Number 1, 2003.

Self-configuring. Systems adapt automatically to dynamically changing environments. When hardware and software systems have the ability to define themselves “on-the fly,” they are self-configuring. This aspect of self-managing systems means that new features, software, and servers can be dynamically added to the enterprise infrastructure with no disruption of services. Systems must be designed to provide this aspect at a feature level with capabilities such as plug and play devices, configuration setup wizards, and wireless server management. These features will allow functions to be added dynamically to the enterprise infrastructure with minimum human intervention. Self-configuring not only includes the ability for each individual system to configure itself in real-time, but also for systems within the enterprise to configure themselves into the on demand Operating Environment. The goal of Autonomic Computing is to provide self-configuration capabilities for the entire IT infrastructure, not just individual servers, software, and storage devices.

Self-healing. Systems discover, diagnose, and react to disruptions. For a system to be self-healing, it must be able to recover from a failed component by first detecting and isolating the failed component, taking it off line, fixing or isolating the failed component, and reintroducing the fixed or replacement component into service without any apparent application disruption. Systems will need to predict problems and take actions to prevent the failure from having an impact on applications. The self-healing objective must be to minimize all outages in order to keep enterprise applications up and available at all times. Developers of system components need to focus on maxi-





mizing the reliability and availability design of each hardware and software product toward continuous availability.

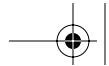
Self-optimizing. Systems monitor and tune resources automatically. Self-optimization requires hardware and software systems to efficiently maximize resource utilization to meet end-user needs without human intervention. IBM systems already include industry-leading technologies such as logical partitioning, dynamic workload management, and dynamic server clustering. These kinds of capabilities should be extended across multiple heterogeneous systems to provide a single collection of computing resources that could be managed by a “logical” workload manager across the enterprise. Resource allocation and workload management must allow dynamic redistribution of workloads to systems that have the necessary resources to meet workload requirements. Similarly, storage, databases, networks, and other resources must be continually tuned to enable efficient operations even in unpredictable environments. Features must be introduced to allow the enterprise to optimize resource usage across the collection of systems within their infrastructure, while also maintaining their flexibility to meet the ever-changing needs of the enterprise.

Basic Level 1	Managed Level 2	Predictive Level 3	Adaptive Level 4	Autonomic Level 5
<ul style="list-style-type: none"> Multiple sources of system generated data Requires extensive, highly skilled staff 	<ul style="list-style-type: none"> Consolidation of data through management tools IT staff analyzes and takes actions 	<ul style="list-style-type: none"> System monitors, correlates, and recommends actions IT staff approves and initiates actions 	<ul style="list-style-type: none"> System monitors, correlates, and takes action IT staff manages performance against SLAs 	<ul style="list-style-type: none"> Integrated components dynamically managed by business rules/policies IT staff focuses on enabling business needs
	<ul style="list-style-type: none"> Greater system awareness Improved productivity 	<ul style="list-style-type: none"> Reduced dependency on deep skills Faster and better decision making 	<ul style="list-style-type: none"> IT agility and resiliency with minimal human interaction 	<ul style="list-style-type: none"> Business policy drives IT management Business agility and resiliency
Manual		Autonomic		

Figure 2 This illustration depicts the 5 transformation levels, from manual to autonomic, evolving to autonomic operations.

Self-protecting. Systems anticipate, detect, identify, and protect themselves from attacks from anywhere. Self-protecting systems must have the ability to





define and manage user access to all computing resources within the enterprise, to protect against unauthorized resource access, to detect intrusions and report and prevent these activities as they occur, and to provide backup and recovery capabilities that are as secure as the original resource management systems. Systems will need to build on top of a number of core security technologies already available today, including LDAP (Lightweight Directory Access Protocol), Kerberos, hardware encryption, and SSL (Secure Socket Layer). Capabilities must be provided to more easily understand and handle user identities in various contexts, removing the burden from administrators.

An Evolution, not a Revolution To implement Autonomic Computing, the industry must take an evolutionary approach and deliver improvements to current systems that will provide significant self-managing value to customers without requiring them to completely replace their current IT environments. New open standards must be developed that will define the new mechanisms for interoperating heterogeneous systems. Figure 2 is a representation of those levels, starting from the basic level, through managed, predictive, and adaptive levels, and finally to the autonomic level.

As seen in Figure 2, the basic level represents the starting point where some IT systems are today. IT professionals who set it up, monitor it, and eventually replace it manage each system element independently. At the managed level, systems management technologies can be used to collect information from disparate systems onto fewer consoles, reducing the time it takes for the administrator to collect and synthesize information as the systems become more complex to operate. In the predictive level, as new technologies are introduced that provide correlation among several elements of the system, the system itself can begin to recognize patterns, predict the optimal configuration, and provide advice on what course of action the administrator should take.

As these technologies improve and as people become more comfortable with the advice and predictive power of these systems, we can progress to the adaptive level where the systems themselves can automatically take the correct actions based on the information that is available to them and the knowledge of what is happening in the systems. Service Level Agreements (SLAs)⁽¹⁶⁾ guide operation of the system. Finally, at the fully autonomic level, the system operation is governed by business policies and objectives. Users interact with the system to monitor the business processes or alter the objectives.

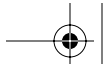




Table 1 Aligning with the on demand business goals

Basic	Managed	Predictive	Adaptive	Autonomic
<i>Process:</i> Informal, reactive, manual	<i>Process:</i> Documented, improved over time, leverage of industry best practices; manual process to review IT performance	<i>Process:</i> Proactive, shorter approval cycle	<i>Process:</i> Automation of many resource mgmt best practices and transaction mgmt best practices, driven by service level agreements	<i>Process:</i> All IT service mgmt and IT resource mgmt best practices are automated
<i>Tools:</i> Local, platform and product specific	<i>Tools:</i> Consolidated resource mgmt consoles with problem mgmt system, automated software install, intrusion detection, load balancing	<i>Tools:</i> Role-based consoles with analysis and recommendations; product configuration advisors; real-time view of current & future IT performance; automation of some repetitive tasks; common knowledge base of inventory and dependency management	<i>Tools:</i> Policy management tools that drive dynamic change based on resource specific policies	<i>Tools:</i> Costing/ financial analysis tools, business and IT modeling tools, tradeoff analysis; automation of some on demand business mgmt roles
<i>Skills:</i> Platform-specific, geographically dispersed with technology	<i>Skills:</i> Multiple skill levels with centralized triage to prioritize and assign problems to skilled IT professionals	<i>Skills:</i> Cross-platform system knowledge, IT workload management skills, some bus process knowledge	<i>Skills:</i> Service objectives and delivery per resource, and analysis of impact on business objectives	<i>Skills:</i> on demand business cost & benefit analysis, performance modeling, advanced use of financial tools for IT context





As companies progress through the five levels of Autonomic Computing, the processes, tools, and benchmarks become increasingly sophisticated, and the skills requirement becomes more closely aligned with the business. The preceding Table illustrates this correlation.

The basic level represents the starting point for most IT organizations. If they are formally measured at all, they are typically measured on the time required to finish major tasks and fix major problems. The IT organization is viewed as a cost center, in which the variable costs associated with labor are preferred over an investment in centrally coordinated systems management tools and processes.

At the managed level, IT organizations are measured on the availability of their managed resources, their time to close trouble tickets in their problem management system, and their time to complete formally tracked work requests. To improve on these measurements, IT organizations document their processes and continually improve them through manual feedback loops and adoption of best practices. IT organizations gain efficiency through consolidation of management tools to a set of strategic platforms and through a hierarchical problem management triage organization.

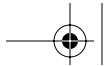
In the predictive level, IT organizations are measured on the availability and performance of their business systems and their return on investment. To improve on these measurements, IT organizations measure, manage, and analyze transaction performance. The implications of the critical nature of the role of the IT organization in the success of the business are understood. Predictive tools are used to project future IT performance, and many tools make recommendations to improve future performance.

In the adaptive level, IT resources are automatically provisioned and tuned to optimize transaction performance. Business policies, business priorities, and service-level agreements guide the autonomic infrastructure behavior. IT organizations are measured on end-to-end business system response times (i.e., transaction performance), the degree of efficiency with which the IT infrastructure is utilized, and their ability to adapt to shifting workloads.

In the autonomic level, IT organizations are measured on their ability to make the business successful. To improve business measurements, IT tools understand the financial metrics associated with on demand business activities and supporting IT activities. Advanced modeling techniques are used to optimize on demand business performance and quickly deploy newly optimized on demand business solutions.

Today's software and hardware system components will evolve toward more autonomic behavior. For example:





- *Data management.* New database software tools can use statistics from the databases, analyze them, and learn from the historical system performance information. The tools can help an enhanced database system automatically detect potential bottlenecks, as they are about to occur and attempt to compensate for them by adjusting tuning parameters. Query optimizers can learn the optimal index and route to certain data and automatically seek out that path based on the historical access patterns and associated response times.
- *Web servers and software.* Web servers can provide real-time diagnostic “dashboard” information, enabling customers to more quickly become aware of resource problems, instead of relying on after-the-fact reports to identify problems. Once improved instrumentation is available, autonomic functions can be introduced that enable the Web server infrastructure to automatically monitor, analyze, and fix performance problems. As an example, suppose an application server is freezing-up intermittently, and no customer transactions are being processed for several seconds, thus losing thousands of dollars in business, as well as customer confidence and loyalty. Using real-time monitoring, predictive analysis, and auto-tuning, the freeze-up is anticipated before it happens. The autonomic function compares real-time data with historical problem data (i.e., suggesting that the cache sizes were set too low). The settings are reset automatically without service disruption, and a report is sent to the administrator that shows what action was taken.
- *Systems management.* Systems management software can contain improved problem determination and data collection features designed to help businesses better diagnose and prevent interruptions (e.g., breaches of security). Such systems management software must enable customers to take an “end-to-end” view of their computing environment across multiple, independently installed hardware and software elements. A bank transaction, for example, might “touch” a discrete database, another transaction, and Web application servers as it is processed across a network. If a problem occurs with processing on one of the individual components, lack of an integrated problem determination infrastructure makes it more difficult to determine what prevented that bank transaction from completing successfully. A consolidated view created by the system management software would enable the system and IT staffs to identify and quickly react to problems as they happen by providing an end-to-end view of the application. The end-to-end view of the environment allows companies to





understand problems and performance information in the context of their business goals.

- *Servers.* Computers can be built that need less human supervision. Computers can try to fix themselves in the event of a failure, protect themselves from hacker attacks, and configure themselves when adding new features. Servers can use software algorithms that learn patterns in Internet traffic or application usage, and provision resources in a way that gives the shortest response time to the task with the highest business priority. Server support for heterogeneous and enterprise workload management, dynamic clustering, dynamic partitioning, improved setup wizards, improved user authentication, directory integration, and other tools to protect access to network resources are all steps toward more autonomic functioning.

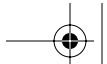
IBM hardware and software systems have already made significant progress in introducing Autonomic Computing functionality.⁽²⁾ However, there is much more work ahead. The efforts to achieve cohesive system behavior must go beyond improvements in the individual components alone. These components must be federated, employing an integrating architecture that establishes the instrumentation, policy, and collaboration technologies so that groups of resources can work in concert, as for example, across systems in a grid. System management tools play a central role in coordinating the actions of system components, providing a simplified mechanism for system administration and for translating business objectives into executable policies to govern the actions of the IT resources available.

Industry Standards Are Needed to Support Autonomic

Computing Most IT infrastructures are composed of components supplied by different vendors. Open industry standards are the key to the construction of Autonomic Computing systems. Systems will need more standardization to introduce a uniform approach to instrumentation and data collection, dynamic configuration, and operation. Uniformity will allow the intersystem exchange of instrumentation and control information to create the basis for collaboration and autonomic behavior among heterogeneous systems.

For example, in storage systems, a standard that has been proposed for specifying data collection items is the Bluefin specification. Bluefin⁽¹⁷⁾ defines a language and schema that allow users to reliably identify, classify, monitor, and control the physical and logical devices in storage area networking. The Storage Networking Industry Association (SNIA) has taken this standard to the Distributed Management Task Force (DMTF). SNIA is using Bluefin as





the basis for its storage management initiative, the intent of which is to become the SNIA standard for management.

In the case of application instrumentation, the standard that has been proposed for obtaining the transaction rate, response time, failure rate, and topology data from applications is the Open Group Application Response Measurement (ARM)⁽¹⁸⁾ application programming interfaces (APIs). The Application Response Measurement API defines the function calls that can be used to instrument an application or other software for transaction monitoring. It provides a way to monitor business transactions by embedding simple cells in the software that can be captured by an agent supporting the ARM API. The calls are used to capture data, allowing software to be monitored for availability, service levels, and capacity.

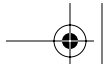
Other standards, such as the DMTF Common Information Model (CIM)⁽¹⁹⁾ and Web Service Level Agreement (WSLA), provide languages and schemas for defining the available data. CIM is an object-oriented information model that provides a conceptual view of physical and logical system components. WSLA is a language to express SLA contracts, to support guaranteed performance, and to handle complex dynamic fluctuations in service demand. SLA-based system management would enable service providers to offer the same Web service at different performance levels, depending on contracts with their customers. WSLA is available through the IBM alphaWorks* Web Services Toolkit⁽²⁰⁾ that features a WSLA document approach based on Extensible Markup Language (XML) to define SLAs.

These standards are technologies that enable the building of “inter-communicating” autonomic system elements that are the foundation for cooperation in a federation of system components. Each individual autonomic “element” is responsible for managing itself, that is, for configuring itself internally, for healing internal failures when possible, for optimizing its own behavior, and for protecting itself from external probing and attack. Autonomic elements are the building blocks for making autonomic systems.

Autonomic elements continuously monitor system (or component) behavior through “sensors” and make adjustments through “effectors.” By monitoring behavior through sensors, analyzing those data, then planning what action should be taken next (if any), and executing that action through effectors, a kind of “control loop”⁽²¹⁾ is created (see the preceding Figure 3).

Interconnecting autonomic elements requires distributed computing mechanisms to access resources across the network. “Grid computing”⁽²²⁾ encompasses the idea of an emerging infrastructure that is focused on networking together heterogeneous, multiple regional and national computing systems.





It has been called the next evolutionary step for the Internet. The term “grid” was chosen as an analogy with the electric power grid, which supplies pervasive access to power. Grids are persistent computing environments that enable software applications to integrate instruments, displays, and computational and information resources that are managed by diverse organizations in widespread locations.

In 2001, the Globus Project^(23, 24) launched a research and development program aimed at creating a toolkit based on the Open Grid Service Architecture (OGSA) that defines standard mechanisms for creating, naming, and discovering services and specifies various protocols to support accessing services. Essentially, OGSA is a framework for distributed computing, based on Web services protocols. Although OGSA is a proposed standard that will be developed and defined in the Global Grid Forum (GGF)⁽²⁵⁾ it is applicable whether the environment consists of a multi-organization grid or simply distributed resources within an enterprise. IBM, Microsoft Corporation, and others have already announced support for the OGSA framework. Work efforts on grid and OGSA are creating important architectural models and new open industry standards that are enablers for the IT industry to make progress toward more self-managing systems. Since grid deployments can expand the domain of computing across many systems, in our view, a successful grid system will require autonomic functionality.

Individual autonomic elements can interact through OGSA mechanisms. For example, today there is no accepted “sensor and effector’s” standard. But, the Globus Toolkit provides information services utilities to provide information about the status of grid resources. One of these utilities is the Monitoring and Discovery Service (MDS).⁽²⁶⁾ MDS 2.2 GRIS “Information Providers” that are essentially sensors or probes. The Globus Toolkit also provides a mechanism for authenticated access to MDS. Fault detection allows a client process to be monitored by a heartbeat monitor. Resource management APIs provide some job management capabilities.

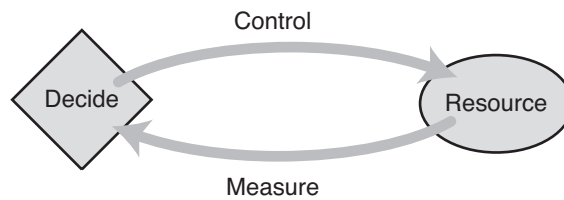
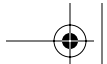


Figure 3 This Illustration depicts the “Control Loop” structure in Autonomic Computing. Source: Autonomic Computing Concepts IBM White Paper, 2001.





Thus we are seeing the emergence of some basic standard mechanisms needed for distributed “control loops” that in turn are needed for Autonomic Computing. When control loop standards are in place, the industry must address the more complex issues of specifying and automating policy management and service level agreements (SLAs).

A typical enterprise has a heterogeneous set of routers, firewalls, Web servers, databases, and workstations, all with different system management mechanisms. So again, industry standards will be needed in order to enable true policy management. We expect that policy specifications will be widely used in enterprises for defining quality of service management, storage backup, and system configuration, as well as security authorization and management.

A common approach to specifying and deploying policy would enable an enterprise to define and disseminate policies that reflect its overall IT service goals. A common, standard set of tools and techniques used throughout the enterprise could simplify analysis and reduce inconsistencies and conflicts in the policies deployed across the various components within the enterprise and also allow a policy exchange with external service providers.

Various standards bodies are working on specifying policies for network and systems management, security, and role-based access control (RBAC). The Internet Engineering Task Force (IETF)⁽²⁷⁾ and DMTF⁽²⁸⁾ have been concentrating on information models for management policies, protocols for transferring policies to network devices, and routing policies; the National Institute of Standards and Technology (NIST)⁽²⁹⁾ is working toward an RBAAutonomic Computing standard; and the Oasis consortium (Organization for the Advancement of Structured Information Standards)⁽³⁰⁾ is working on an XML-based specification of access control policies and authentication information.

It will take some time for the current divergent standards policy-based solutions to come to embrace a common approach. Meanwhile, research on policy-based management approaches continues.^(31,32) Advances in policy management are needed to enable enterprises to eventually specify the behaviors of IT services in terms of the business process objectives of the enterprises.

Exploratory Research and Development Presented in This Issue
{*IBM Systems Journal, Volume 42, No. 1, 2003*} Autonomic Computing represents an exciting new research direction in computing. IBM believes that meeting the grand challenge of Autonomic Computing systems will involve





researchers in a diverse array of fields, including systems management, distributed computing, networking, operations research, software development, storage, artificial intelligence, and control theory, as well as others.

The challenge of Autonomic Computing requires more than the re-engineering of today's systems. Autonomic Computing also requires new ideas, new insights, and new approaches. This issue of the IBM Systems Journal provides just a glimpse into an array of research and development efforts underway for Autonomic Computing. Below we present the topics in the issue.

D. C. Verma, S. Sahu, S. Calo, A. Shaikh, I. Chang, and A. Acharya in their paper, "SRIRAM: A Scalable Resilient Autonomic Mesh,"⁽³³⁾ propose a method that facilitates instantiating mirroring and replication of services in a network of servers.

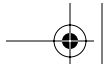
The ability to redistribute hardware resources dynamically is essential to both the self-configuring and self-optimizing goals of Autonomic Computing. J. Jann, L. M. Browning, and R. S. Burugula describe this new server capability in "Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers."⁽³⁴⁾

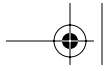
In the first of two invited papers, D. A. Norman and A. Ortony from Northwestern University, along with D. M. Russell of IBM, discuss in "Affect and Machine Design: Lessons for the Development of Autonomous Machines"⁽³⁵⁾ how studying the human characteristics of cognition and affect will help designers in developing complex autonomic systems that will interact with unpredictable situations.

K. Whisnant, Z. T. Kalbarczyk, and R. K. Iyer examine the difficulties of dynamically reconfiguring application software in their paper, "A System Model for Dynamically Reconfigurable Software."⁽³⁶⁾ They believe that both static structure and runtime behaviors must be captured in order to define a workable reconfiguration model.

One technology to support self-healing and self-configuring is the ability to dynamically insert new pieces of software and remove other pieces of code, without shutting down the running system. This technology is being explored in the K42 research operating system and is presented in the paper by J. Appavoo, K. Hui, C. A. N. Soules, R. W. Wisniewski, D. M. Da Silva, O. Krieger, M. A. Auslander, D. J. Edelson, B. Gamsa, G. R. Ganger, P. McKenney, M. Ostrowski, B. Rosenberg, M. Stumm, and J. Xenidis, entitled "Enabling Autonomic Behavior in Systems Software with Hot Swapping."⁽³⁷⁾

L. W. Russell, S. P. Morgan, and E. G. Chron introduce the idea of a predictive autonomic system in their paper entitled "Clockwork: A New Move-





ment in Autonomic Systems.”⁽³⁸⁾ They explore the idea of a system that anticipates workload needs based on statistical modeling, tracking, and forecasting.

Component-based development, where multiple distributed software components are composed to deliver a particular business function, is an emerging programming model in the Web services world. D. M. Yellin in his paper, “Competitive Algorithms for the Dynamic Selection of Component Implementations,”⁽³⁹⁾ proposes a strategy and framework for optimizing component performance based on switching between different component implementations.

In an example of “self-optimizing,” V. Markl, G. M. Lohman, and V. Raman discuss improving query performance by comparing estimates with actual results toward self-validating query planning in “LEO: An Autonomic Query Optimizer for DB2.”⁽⁴⁰⁾

As noted, system and network security are fundamental to Autonomic Computing systems. In “Security in an Autonomic Computing Environment,”⁽⁴¹⁾ D. M. Chess, C. C. Palmer, and S. R. White outline a number of security and privacy issues in the design and development of autonomic systems.

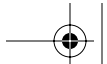
G. Lanfranchi, P. Della Peruta, A. Perrone, and D. Calvanese describe what they see as a paradigm shift in system management needed for Autonomic Computing. In their paper, “Toward a New Landscape of Systems Management in an Autonomic Computing Environment,”⁽⁴²⁾ they introduce a knowledge-based resource model technology that extends across design, delivery, and run time.

In the second invited paper, “Comparing Autonomic and Proactive Computing,”⁽⁴³⁾ R. Want, T. Pering, and D. Tennenhouse of Intel Research present a high-level discussion of the similarities between proactive computing and Autonomic Computing with an emphasis on their research in proactive computing—an environment in which computers anticipate what users need and act accordingly.

Today, optimizing performance in multisystem e-commerce environments requires considerable skill and experience. In “Managing Web Server Performance with AutoTune Agents,”⁽⁴⁴⁾ Y. Diao, J. L. Hellerstein, S. Parekh, and J. P. Bigus describe intelligent agents that use control theory techniques to automatically adjust an Apache** Web server to dynamic workloads.

The backbone of a grid or typical Autonomic Computing system is an intelligent, heterogeneous network infrastructure. Management issues related to topology, R. Haas, P. Droz, and B. Stiller in “Autonomic Service Deployment





in Networks” explore service placement, cost and service metrics, as well as dynamic administration structure.⁽⁴⁵⁾

Although much of the discussion on Autonomic Computing often focuses on servers, networks, databases, and storage management, we realize that personal computer users would also benefit greatly by the introduction of autonomic features. D. F. Bantz, C. Bisdikian, D. Challener, J. P. Karidis, S. Mastrianni, A. Mohindra, D. G. Shea, and M. Vanover explore these possibilities in their paper, “Autonomic Personal Computing.”⁽⁴⁶⁾

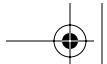
People will still need to interact with Autonomic Computing systems. D. M. Russell, P. P. Maglio, R. Dordick, and C. Neti in their paper entitled “Dealing with Ghosts: Managing the User Experience of Autonomic Computing”⁽⁴⁷⁾ argue that the lessons we have learned in human-computer interaction research must be applied to effectively expose and communicate the run-time behavior of these complex systems and to better define and structure the user system operation scenarios.

In the Technical Forum section, the complex challenges of life-cycle management and providing capacity On Demand are examined in a project as described by A. Abbondanzio, Y. Aridor, O. Biran, L. L. Fong, G. S. Goldszmidt, R. E. Harper, S. M. Krishnakumar, G. Pruett, and B.-A. Yassur in “Management of Application Complexes in Multitier Clustered Systems.”⁽⁴⁸⁾

Topic Conclusion In his keynote speech at the Almaden Institute 2002,⁽⁴⁹⁾ John Hennessy, President of Stanford University, presented his view of the autonomic challenge. While acknowledging the significant accomplishments in hardware architecture over the past 20 years, he urged industry and academia to look forward and to shift focus to a set of issues related to how services will be delivered over networks in the Internet/Web-centric “post-desktop” era: “As the business use of this environment grows and as people become more and more used to it, the flakiness that we’ve all accepted in the first generation of the Internet and the Web—will become unacceptable.” Hennessy emphasized an increased research focus on availability, maintainability, scalability, cost, and performance—all fundamental aspects of Autonomic Computing.

Autonomic computing is a journey. Progress will be made in a series of evolutionary steps. This {referenced} issue of the *IBM Systems Journal* presents some of the technology signposts that can serve to guide the ongoing research in this new direction.





“The Dawning” Acknowledgments The authors of “The Dawning” express thanks to the many authors who submitted their work for publication. Special thanks go to Lorraine Herger, Kazuo Iwano, Pratap Pattnaik, Connie Marconi, and Felicia C. Medley, who organized the call for papers, managed the process to select the topics and the papers, and worked with the authors and *IBM Systems Journal* staff to produce this issue. We also thank the numerous referees whose comments helped all the authors in improving their papers. We particularly express our appreciation to Paul Horn, IBM Senior Vice President and Director of Research, for launching the Autonomic Computing grand challenge in his 2001 presentation on Autonomic Computing and to Irving Wladawsky-Berger who led the launch of the Autonomic Computing project in the IBM Server Group. We acknowledge the efforts of Robert Morris, Anant Jhingran, Tushar Chandra, and K. M. Moiduddin, who organized the Almaden Autonomic Computing Institute held at San Jose, California in April 2002. We also acknowledge Sam S. Adams, Lisa F. Spainhower, William H. Tetzlaff, William H. Chung, Steve R. White, and Kazuo Iwano who organized the Autonomic Computing Conference sponsored by the IBM Academy of Technology and held in Yorktown Heights, New York in May 2002. Our thanks go to Christopher W. Luongo for his article, “Server to IT: Thanks, But I Fixed It Myself.” The authors would like to thank Ric Telford, John Sweitzer, and Jim Crosskey of the Autonomic Computing department for their contributions to this manuscript. The authors also wish to acknowledge the IBM Systems Journal staff for their many helpful suggestions during the creation of this issue.

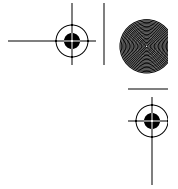
* Trademark or registered trademark of International Business Machines Corporation.

** Trademark or registered trademark of Apache Digital Corporation.

Cited References and Notes

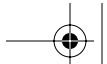
1. P. Horn, *Autonomic Computing: IBM’s Perspective on the State of Information Technology*, IBM Corporation (October 15, 2001); available at http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.
2. IBM Server Group, *eLiza: Building an Intelligent Infrastructure for E-business—Technology for a Self-Managing Server Environment*, G520-9592-00, IBM Corporation (2001); also at http://www-1.ibm.com/servers/eserver/introducing/eliza/eliza_final.pdf.





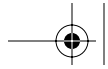
3. For more than 30 years, Moore's Law has forecasted progress in the computing industry like an immutable force of nature. In 1965, Gordon E. Moore, then the director of research and development at Fairchild Semiconductor Corporation, made the casual observation that processing power will double every 18 to 24 months, suggesting healthy growth ahead over the next decade for the then-nascent silicon chip industry. Five years later, Moore co-founded Intel Corporation, and "Moore's law" was well on its way to becoming a self-fulfilling prophecy among researchers and developers in the industry.
4. A petabyte (PB) is 1000 terabytes, and a terabyte is 1000 gigabytes. CERN, the European Center for Nuclear Research located just outside of Geneva, Switzerland, has started building a 100-PB archive to house data from the particle accelerator of the research center. See "From Kilobytes to Petabytes in 50 Years," *Science and Technology Review*, UCRL-52000-02-4, Lawrence Livermore National Laboratory, Livermore, CA (April 15, 2002), <http://www.llnl.gov/str/March02/March50th.html>.
5. The term "Autonomic Computing" comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations without any external help. "Autonomic" is defined as: (a) Of, relating to, or controlled by the autonomic nervous system; (b) Acting, or occurring involuntarily; automatic; an autonomic reflex.
6. F. P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*, Twentieth Anniversary Edition, Addison-Wesley Publishing Co., Reading, MA (1995), p. 226. See also, F. P. Brooks, Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer* 20, No. 4, 1019 (1987).
7. D. A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, N. Treuhaft, *Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies*, U.C. Berkeley Computer Science Technical Report, UCB//CSD-02-1175, University of California, Berkeley (March 15, 2002).
8. K. Evans-Correia, "Simplifying Storage Management Starts with More Efficient System Utilization," Interview with N. Tabellion, *searchStorage* (August 29, 2001), see http://searchstorage.techtarget.com/qna/0,289202,sid5_gci764063,00.html.
9. *IBM Data Management Tools: New Opportunities for Cost-Effective Administration*, Profile Report, Aberdeen Group, Inc., Boston (April 2002), p. 3.





10. D. Patterson, "Availability and Maintainability Performance: New Focus for a New Century," *USENIX Conference on File and Storage Technologies (FAST '02)*, Keynote Address, Monterey, CA (January 29, 2002).
11. A. Brown and D. A. Patterson, "To Err Is Human," *Proceedings of the First Workshop on Evaluating and Architecting System Dependability (EASY '01)*, Goeteborg, Sweden (July 2001).
12. "How Much Is an Hour of Downtime Worth to You?" from *Must-Know Business Continuity Strategies*, Yankee Group, Boston (July 31, 2002).
13. D. J. Clancy, "NASA Challenges in Autonomic Computing," *Almaden Institute 2002*, IBM Almaden Research Center, San Jose, CA (April 10, 2002).
14. Merit Project, Computer Associates International, http://www.merit-project.com/it_survey_results.htm.
15. I. Wladawsky-Berger, "Advancing E-business into the Future: The Grid," *Kennedy Consulting Summit 2001*, New York (November 29, 2001).
16. In this context, a Service Level Agreement (SLA) is a compact between a customer or consumer and a provider of an IT service that specifies the levels of availability, serviceability, performance (and tracking/reporting), problem management, security, operation, or other attributes of the service, often established via negotiation. Typically, an SLA identifies and defines the customer's needs, provides a framework for discussion and understanding, attempts to simplify complex requirements, outlines methods for resolving disputes, and helps eliminate unrealistic expectations.
17. "Bluefin' A Common Interface for SAN Management," White Paper, Storage Networking Industry Association (August 13, 2002), http://www.snia.org/tech_activities/SMI/bluefin/Bluefin_White_Paper_v081302.pdf from http://www.snia.org/tech_activities/SMI/bluefin/.
18. *Application Response Measurement Issue 3.0 Java Binding*, Open Group Technical Standard CO14, The Open Group (October 2001), at <http://www.opengroup.org/products/publications/catalog/c014.htm>.
19. *Common Information Model (CIM) Specification Version 2.2*, DSP0004, Distributed Management Task Force (June 14, 1999), at http://www.dmtf.org/standards/standard_cim.php.
20. Web Services Toolkit, alphaWorks, IBM Corporation (July 26, 2000), http://www.alphaworks.ibm.com/tech/webservices_toolkit.
21. The control loop is the essence of automation. By measuring or sensing some activity in a process to be controlled, a controller component

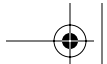




decides what needs to be done next and executes the required operations through a set of actuators. The controller then re-measures the process to determine whether the actions of the actuator had the desired effect. The whole routine is then repeated in a continuous loop of measure, decide, actuate, and repeat.

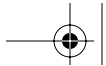
22. *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Editors, Morgan Kaufmann Publishers, Inc., San Francisco, CA (1999).
23. See <http://www.globus.org>, the home of the Globus Project, the Globus Toolkit (GT3), and work related to the Open Grid Services Architecture (OGSA).
24. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing* 15, No. 3, 200222 (2001); see also, <http://www.globus.org/research/papers/anatomy.pdf>.
25. Global Grid Forum, <http://www.gridforum.org/>.
26. *MDS 2.2 User's Guide*, The Globus Project, available at www.globus.org/mds/mdsusersguide.pdf.
27. Internet Engineering Task Force, <http://www.dmtf.org>.
28. Distributed Management Task Force, <http://www.dmtf.org>.
29. National Institute of Standards and Technology, <http://www.nist.org>.
30. Organization for the Advancement of Structured Information Standards, <http://www.oasis-open.org>.
31. L. Lymberopoulos, E. Lupu, and M. Sloman, "An Adaptive Policy Based Management Framework for Differentiated Services Networks," *Proceedings of the 3rd IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, Monterey, CA (June 2002), pp. 147158.
32. V. Sander, W. A. Adamson, I. Foster, and A. Roy, "End-to-End Provision of Policy Information for Network QoS," *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press (August 2001).
33. D. C. Verma, S. Sahu, S. Calo, A. Shaikh, I. Chang, and A. Acharya, "SRIRAM: A Scalable Resilient Autonomic Mesh," *IBM Systems Journal* 42, No. 1, 1928 (2003, this issue).
34. J. Jann, L. A. Browning, and R. S. Burugula, "Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers," *IBM Systems Journal* 42, No. 1, 2937 (2003, this issue).





35. D. A. Norman, A. Ortony, and D. M. Russell, "Affect and Machine Design: Lessons for the Development of Autonomous Machines," *IBM Systems Journal* **42**, No. 1, 3844 (2003, this issue).
36. K. Whisnant, Z. T. Kalbarczyk, and R. K. Iyer, "A System Model for Dynamically Reconfigurable Software," *IBM Systems Journal* **42**, No. 1, 4559 (2003, this issue).
37. J. Appavoo, K. Hui, C. A. N. Soules, R. W. Wisniewski, D. M. Da Silva, O. Krieger, M. A. Auslander, D. J. Edelson, B. Gamsa, G. R. Ganger, P. McKenney, M. Ostrowski, B. Rosenberg, M. Stumm, and J. Xenidis, "Enabling Autonomic Behavior in Systems Software with Hot Swapping," *IBM Systems Journal* **42**, No. 1, 6076 (2003, this issue).
38. L. W. Russell, S. P. Morgan, and E. G. Chron, "Clockwork: A New Movement in Autonomic Systems," *IBM Systems Journal* **42**, No. 1, 7784 (2003, this issue).
39. D. M. Yellin, "Competitive Algorithms for the Dynamic Selection of Component Implementations," *IBM Systems Journal* **42**, No. 1, 8597 (2003, this issue).
40. V. Markl, G. M. Lohman, and V. Raman, "LEO: An Autonomic Query Optimizer for DB2," *IBM Systems Journal* **42**, No. 1, 98106 (2003, this issue).
41. D. M. Chess, C. C. Palmer, and S. R. White, "Security in an Autonomic Computing Environment," *IBM Systems Journal* **42**, No. 1, 107118 (2003, this issue).
42. G. Lanfranchi, P. Della Peruta, A. Perrone, and D. Calvanese, "Toward a New Landscape of Systems Management in an Autonomic Computing Environment," *IBM Systems Journal* **42**, No. 1, 119128 (2003, this issue).
43. R. Want, T. Pering, and D. Tennenhouse, "Comparing Autonomic and Proactive Computing," *IBM Systems Journal* **42**, No. 1, 129135 (2003, this issue).
44. Y. Diao, J. L. Hellerstein, S. Parekh, and J. P. Bigus, "Managing Web Server Performance with AutoTune Agents," *IBM Systems Journal* **42**, No. 1, 136149 (2003, this issue).
45. R. Haas, P. Droz, and B. Stiller, "Autonomic Service Deployment in Networks," *IBM Systems Journal* **42**, No. 1, 150164 (2003, this issue).
46. D. F. Bantz, C. Bisdikian, C. Challener, J. P. Karidis, S. Matrianni, A. Mohindra, D. G. Shea, and M. Vanover, "Autonomic Personal Computing," *IBM Systems Journal* **42**, No. 1, 165176 (2003, this issue).





47. D. M. Russell, P. P. Maglio, R. Dordick, and C. Neti, "Dealing with Ghosts: Managing the User Experience of Autonomic Computing," *IBM Systems Journal* **42**, No. 1, 177188 (2003, this issue).
48. A. Abbondanzio, Y. Aridor, O. Biran, L. L. Fong, G. S. Goldszmidt, R. E. Harper, S. M. Krishnakumar, G. Pruett, and B.-A. Yassur, "Management of Application Complexes in Multitier Clustered Systems," Technical Forum, *IBM Systems Journal* **42**, No. 1, 189195 (2003, this issue).
49. J. Hennessy, "Back to the Future: Time to Return to Some Long-Standing Problems in Computer Science," *Almaden Institute 2002*, IBM Almaden Research Center, San Jose, CA (April 10, 2002).

An Architectural Blueprint for Autonomic Computing

This section provides full treatment of the IBM strategic perspectives in Autonomic Computing. The discussions presented in this section will describe the strategic *Autonomic Computing Blueprint*. This Autonomic Computing Blueprint [ACBLUEPRINT] is developed, published, and maintained by the IBM Autonomic Computing Group located in the IBM Thomas J. Watson Research Center in Hawthorne, New York. This architectural Autonomic Computing Blueprint is one of the key instruments to understanding the on demand business strategy perspectives.

WHAT IS AUTONOMIC COMPUTING?

The term "*Autonomic Computing*" is derived and paralleled from the body's autonomic nervous system. The same way we take for granted the human body's capabilities for the management of breathing, digestion, and fending off viruses, companies will one day take for granted the network's ability to manage, repair, and protect itself from an enterprise perspective.

In this stage of on demand business, realizing autonomic capabilities will bring a totally new kind of transformation—or, more specifically, new levels of integration: of processes and applications inside the business; of suppliers and distributors at either end of the business; of customers outside the enterprise and of employees inside it. Until now, companies have been "on the 'Net." The on demand business transformation will now place companies in such a way that they will become an integrated part of the 'Net.

The high-tech industry has spent decades creating computer systems with ever-mounting degrees of complexity to solve a wide variety of business problems. Ironically, complexity itself has become part of the problem. As networks and distributed systems grow and change, they can become increasingly hampered by system deployment failures and hardware and software issues, not to mention human error. Such scenarios in turn require further



human intervention to enhance the performance and capacity of IT components. This drives up the overall IT costs—even though technology component costs continue to decline. As a result, many IT professionals seek ways to improve the ROI in their IT infrastructure, by reducing the total cost of ownership (TCO) of their environments while improving the QoS for users.

We do not see a slowdown in Moore's law as the main obstacle to further progress in the IT industry. Rather, it is our industry's exploitation of the technologies that has arisen in the wake of Moore's law that has led us to the verge of a complexity crisis. Software developers have fully exploited a four to six order-of-magnitude increase in computational power—producing ever more sophisticated software applications and environments. There has been exponential growth in the number and variety of systems and components. The value of database technology and the Internet has fueled significant growth in storage subsystems, which are now capable of holding petabytes of structured and unstructured information. Networks have interconnected our distributed, heterogeneous systems. Our information society creates unpredictable and highly variable workloads on those networked systems. And today, those increasingly valuable, complex systems require more and more skilled IT professionals to install, configure, operate, tune, and maintain them.

Autonomic Computing helps address these complexity issues by using technology to manage technology. The idea is not new—many of the major players in the industry have developed and delivered products based on this concept.

The term “autonomic” is derived from human biology. The autonomic nervous system monitors your heartbeat, checks your blood sugar level, and keeps your body temperature close to 98.6°F without any conscious effort on your part. In much the same way, Autonomic Computing components anticipate computer system needs and resolve problems—with minimal human intervention. Figure 3.9 shows this computer intersection.



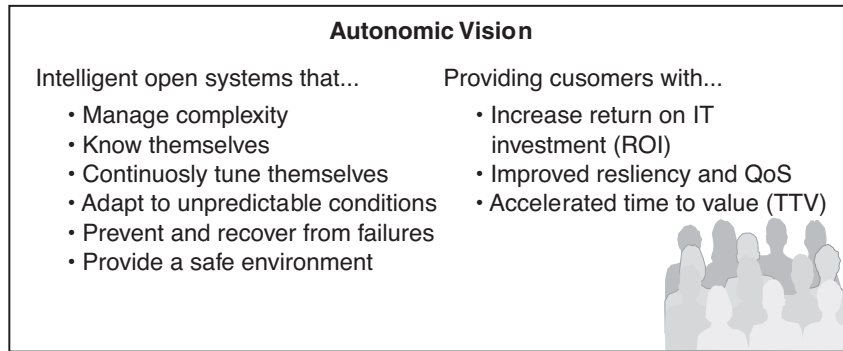
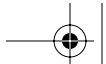


FIGURE 3.9 The vision for Autonomic Computing incorporates “intelligent” open systems and important customer values.

However, there is an important distinction between autonomic activity in the human body and autonomic responses in computer systems. Many of the decisions made by autonomic elements in the body are involuntary, whereas autonomic elements in computer systems make decisions based on tasks you choose to delegate to the technology. In other words, adaptable policy—rather than rigid hard-coding—determines the types of decisions and actions autonomic elements make in computer systems.

Autonomic Computing can result in significant improvements in system management efficiency when the disparate technologies that manage the environment work together to deliver performance results system-wide. For this to be possible in a multi-vendor infrastructure, however, IBM and other vendors must agree on a common approach to architecting autonomic systems.

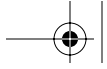
The Customer Value of Autonomic Computing

An on demand business enterprise is one whose business processes—integrated end-to-end across the company and with key partners, suppliers, and customers—can respond with agility and speed to any customer demand, market opportunity, or external threat.

To realize the benefits of on demand business, customers will need to embrace a new computing architecture that allows them to best leverage existing assets as well as those that lie outside traditional corporate boundaries. This on demand Operating Environment has four essential characteristics: It is integrated, open, virtualized, and autonomic.

Autonomic Computing was conceived as a way to help reduce the cost and complexity of owning and operating an IT infrastructure. In an autonomic





environment, system components—from hardware such as desktop computers and mainframes to software such as operating systems and business applications—are self-configuring, self-healing, self-optimizing, and self-protecting.

These self-managing attributes are at the core of an Autonomic Computing environment. They suggest that the tasks involved in configuring, healing, optimizing, and protecting the IT system are initiated due to situations the technologies themselves detect, and that these tasks are performed by those same technologies. Collectively, these intuitive and collaborative characteristics enable enterprises to operate efficiently with fewer human resources, while decreasing costs and enhancing the organization's ability to react to change. For instance, in a self-managing system, a new resource is simply deployed and then optimization occurs. This is a notable shift from traditional implementations, in which a significant amount of analysis is required before deployment to ensure that the resource will run effectively.

Finally, it is important to be aware that the self-configuring, self-healing, self-optimizing, and self-protecting attributes are not independent of one another. Specifically, all four attributes allow the ability to make changes to any configuration of one or more aspects of the IT system. The motivation for the configuration change is different for each attribute.



The Autonomic Computing Blueprint

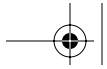
The architectural Autonomic Computing Blueprint (ACBLUEPRINT, hereinafter simply referred to as the “blueprint”) is an overview of the basic strategic perspectives, architectural concepts, technological constructs, and behaviors for building autonomic capabilities into on demand Operating Environments.

The blueprint also describes, in a concise and hard-hitting manner, the initial set of core capabilities for enabling Autonomic Computing, and it discusses the technologies that support these core capabilities. The blueprint also discusses industry standards, emerging standards, and new areas for standardization that will deliver Autonomic Computing open system architectures.

Autonomic Computing Architectural Concepts

The architectural concepts presented in this section begin the process of developing a common approach and terminology to architecting Autonomic Computing systems. The Autonomic Computing architecture concepts provide a mechanism for discussing, comparing, and contrasting the





approaches different vendors use to deliver self-managing attributes in an Autonomic Computing system. The Autonomic Computing architecture starts from the premise that implementing self-managing attributes involves an intelligent control loop. This loop collects information from the system, makes decisions, and then adjusts the system as necessary. An intelligent control loop can enable the system to do things such as:

- Self-configure, by installing software when it detects that software is missing
- Self-heal, by restarting a failed element
- Self-optimize, by adjusting the current workload when it observes an increase in capacity
- Self-protect, by taking resources offline if it detects an intrusion attempt

Figure 3.10 illustrates that these control loops can be delivered in two different ways:

- Various combinations of management tools or products can implement a loop. In Figure 3.10, the three examples are the configuration manager, workload manager, and risk manager. These tools use the instrumentation interfaces (for example, a Simple Network Management Protocol Management Information Base [SNMP MIB]) provided by IT system components to make the control loop manageable. This interface is referred to as the “manageability interface” in the figure.
- A control loop, which embeds a loop in the runtime environment for a particular resource, can be provided by a resource provider. In this case, the control loop is configured through the manageability interface provided for that resource (for example, a hard drive). In some cases, the control loop may be hard-wired or hard-coded so it is not visible through the manageability interface.



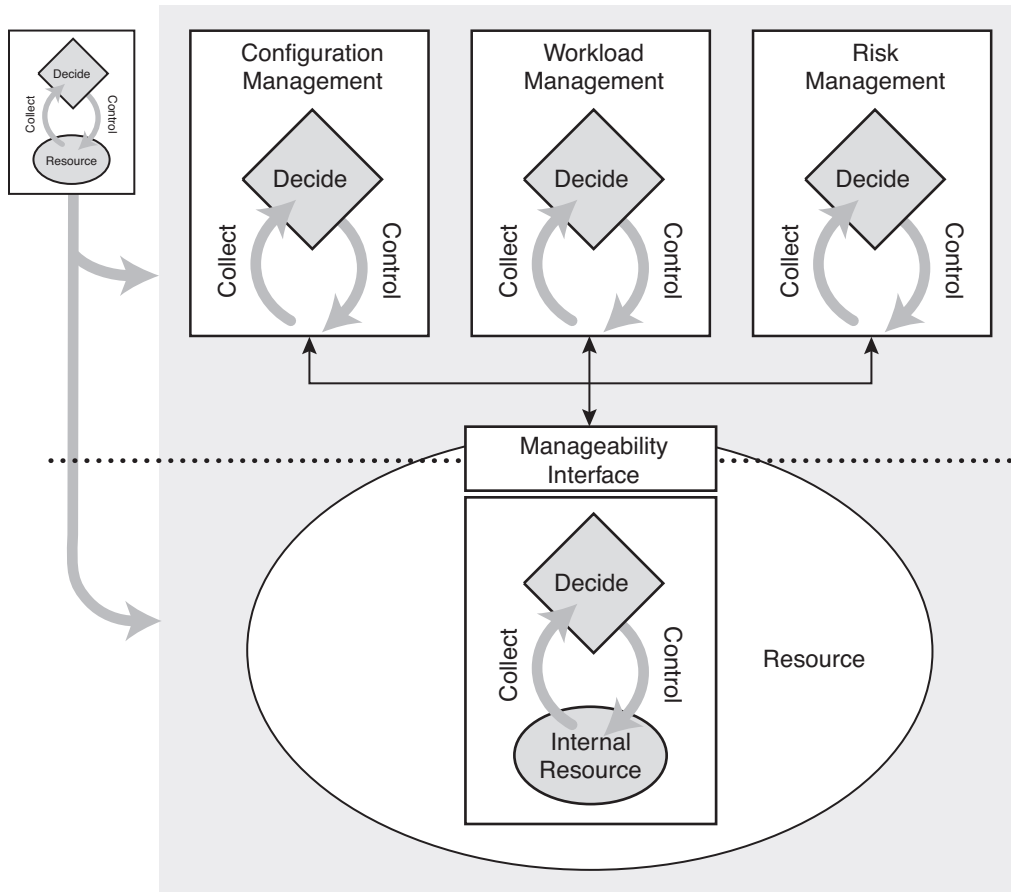


FIGURE 3.10 The flow of a control loop. A control loop can be delivered in two ways: in management tools and in embedded system resources.

Decision-Making Contexts

In the previous figure, three management functions were shown: configuration management, workload management, and risk management. Each of these management functions implements a different control loop, but they can potentially interact with the same resource. Thus, it is possible to have multiple control loops managing the same resource. In general, a robust IT system can have thousands of active control loops at any point in time.

To provide some order to this situation, the architecture for Autonomic Computing defines three different layers of management. Each layer



involves implementing control loops to enable self-management in different decision-making contexts, or scopes:

- The resource element context is the most basic because its elements—networks, servers, storage devices, applications, middleware, and personal computers—manage themselves in an autonomic environment. The resource element layer is where autonomic function begins, by having intelligent control loops that configure, optimize, heal, and protect individual resources.
- Resource elements are grouped into a composite resources decision-making context. A pool of servers that work together to dynamically adjust workload and configuration to meet certain performance and availability thresholds can represent these groups; or, they can be represented by a combination of heterogeneous devices, such as databases, Web servers, and storage subsystems, which work together to achieve common performance and availability targets.

These different management levels define a set of decision-making contexts that are used to classify the purpose and role of a control loop within the Autonomic Computing architecture.

Control Loop Structure

In an autonomic environment, components work together, communicating with each other and with high-level management tools. They regulate themselves, and sometimes, each other. They can proactively manage the system, while hiding the inherent complexity of these activities from end-users and IT professionals.

Another aspect of the Autonomic Computing architecture is shown in the Figure 3.11. This portion of the architecture details the functions that can be provided for the control loops. The architecture organizes the control loops into two major elements: a managed element and an autonomic manager. A managed element is what the autonomic manager is controlling. An autonomic manager is a component that implements a particular control loop.



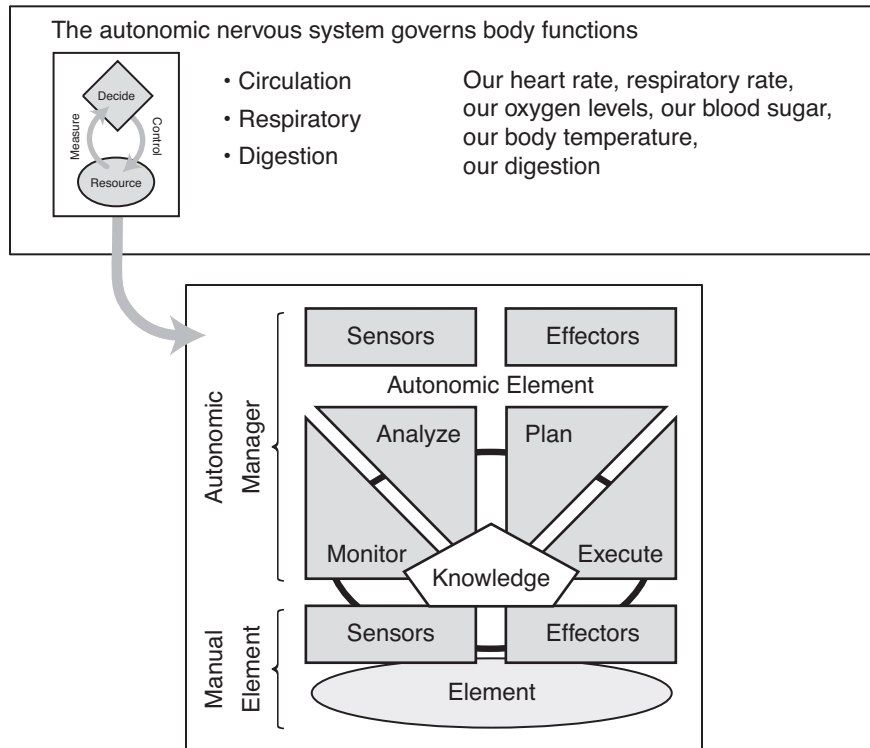
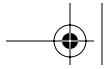


FIGURE 3.11 In an Autonomic Computing architectures, control loops facilitate system management.

Managed Elements

A *managed element* is a controlled system component. A managed element can be a single resource (a server, database server, or router) or a collection of resources (a pool of servers, cluster, or business application). A managed element is controlled through its *sensors* and *effectors*:

- Sensors provide mechanisms to collect information about the state and state transition of an element. To implement sensors, you can either use a set of “get” operations to retrieve information about the current state, or a set of management events (unsolicited, asynchronous messages or notifications) that flow when the state of the element changes in a significant way.
- Effectors are mechanisms that change the state (configuration) of an element. In other words, effectors are a collection of “set” commands or application programming interfaces (APIs) that change the configuration of the managed resource in some important way.



The combination of sensors and effectors forms the manageability interface that is available to an autonomic manager. As depicted by the black lines in the figure that connect the elements on the sensors and effectors, the architecture encourages the idea that sensors and effectors are linked together. For example, a configuration change that occurs through effectors should be reflected as a configuration change notification through the sensor interface.

Autonomic Manager

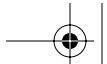
The *autonomic manager* is the component that implements the control loop. The architecture dissects the loop into four parts that share knowledge:

- The *monitor part* provides the mechanisms that collect, aggregate, filter, manage, and report details (metrics and topologies) collected from an element.
- The *analyze part* provides the mechanisms to correlate and model complex situations (time-series forecasting and queuing models, for example). These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
- The *plan part* provides the mechanisms to structure the action needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
- The *execute part* provides the mechanisms that control the execution of a plan with considerations for on-the-fly updates.

Some autonomic elements will have as their main task the management of an IT resource, such as a DB2 information management system from IBM, a Linux Web server, an IBM TotalStorage Enterprise Storage Server storage array, or a network router or load balancer. The autonomic element's autonomic manager will make every effort to carry out the task as efficiently as possible, based on high-level policies that govern the apportionment of the resource, or specify who is to have access to it or place constraints on how the resource is to be made available. The autonomic manager relies on techniques such as feedback control optimization based on forecasting models.

This architecture does not prescribe a particular management protocol or instrumentation technology since the architecture needs to work with the various computing technologies and standards that exist in the industry today, such as SNMP, JavaManagement Extensions (JMX), Distributed Management Task Force, Inc. (DMTF), Common Information Model (CIM), vendor-specific APIs or commands, as well as any new technologies that emerge in the future. Given the diversity of the approaches that already exist in the IT industry, this architecture endorses Web services techniques for sensors





and effectors. These techniques encourage implementers to leverage existing approaches and support multiple binding techniques as well as multiple marshalling techniques.

Figure 3.12 provides a more detailed view of these four parts by highlighting some of the functions each part uses:

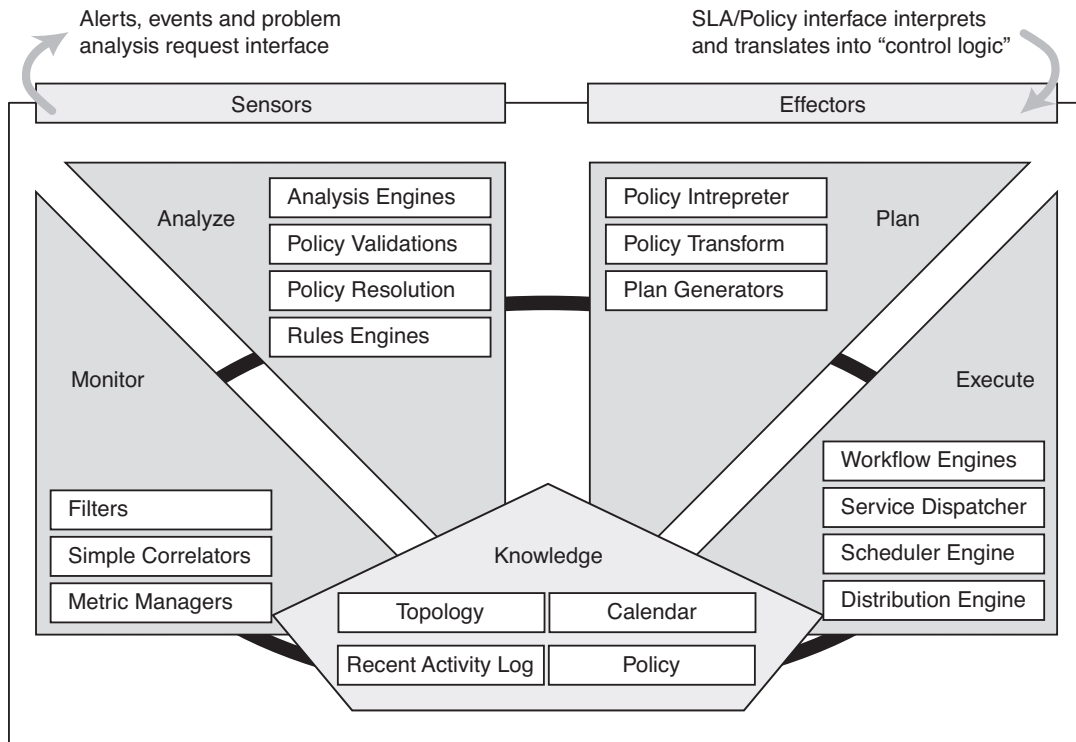


FIGURE 3.12 The functional details of an autonomic manager.

These four parts work together to provide control loop functionality. The diagram shows a structural arrangement of the parts—not a control flow. The bold line that connects the four parts should be thought of as a common messaging bus rather than a strict control flow. In other words, there can be situations where the plan part may ask the monitor part to collect more or less information. There could also be situations where the monitor part may trigger the plan part to create a new plan. The four parts collaborate using asynchronous communication techniques, like a messaging bus.





Autonomic Manager Collaboration

The numerous autonomic managers in a complex IT system must work together to deliver Autonomic Computing to achieve common goals. For example, a database system needs to work with the server, storage sub-system, storage management software, Web server, and other elements of the system for the IT infrastructure as a whole to become a self-managing system.

The sensors and effectors provided by the autonomic manager facilitate collaborative interaction with other autonomic managers. In addition, autonomic managers can communicate with each other in both P2P and hierarchical arrangements.

Figure 3.13 shows an example of a simple IT system that includes two business applications: a customer order application and a vendor relationship application. Separate teams manage these applications. Each of these applications depends on a set of IT resources—databases and servers—to deliver its functionality. Some of these resources—DB 3, DB 4, Server B, and Server C—are shared between the applications, which are managed separately.

There is a minimum of four management domains (decision-making contexts) in this example. Each of the applications (customer order and vendor relationship) has a domain that is focused on the business system it implements. In addition, there is a composite resource domain for managing the common issues across the databases and a composite resource domain for managing common issues for the servers.



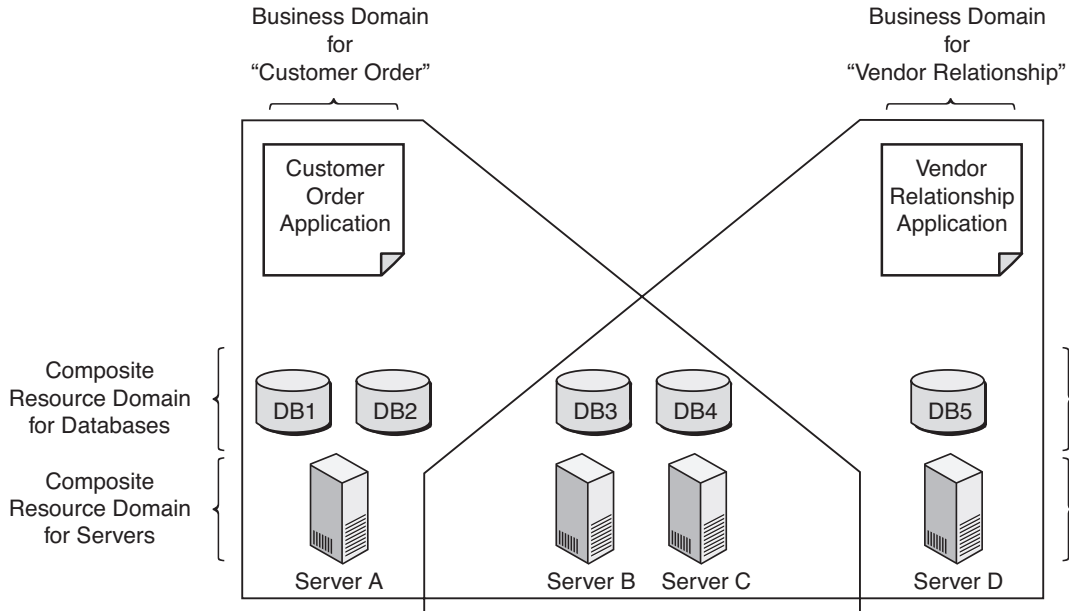


FIGURE 3.13 IT systems can share resources to increase efficiency.

Now, let us apply the Autonomic Computing architecture to this example, to see how autonomic managers would be used. Figure 3.14 illustrates some of the autonomic managers that either directly or indirectly manage DB 3 and some of the interaction between these autonomic managers. There are six autonomic managers in this illustration: one for each of the management domains, one embedded in the DB 3 resource, and one dedicated to the specific database resource.

Since the decision-making contexts for these autonomic managers are interdependent and self-optimizing, the autonomic managers for the various contexts will need to cooperate. This is accomplished through the sensors and effectors for the autonomic managers, using a “matrix management protocol.” This protocol makes it possible to identify situations in which there are “multiple managers,” and enables autonomic managers to electronically negotiate resolutions for domain conflicts based on a system-wide business and resource optimization policy.

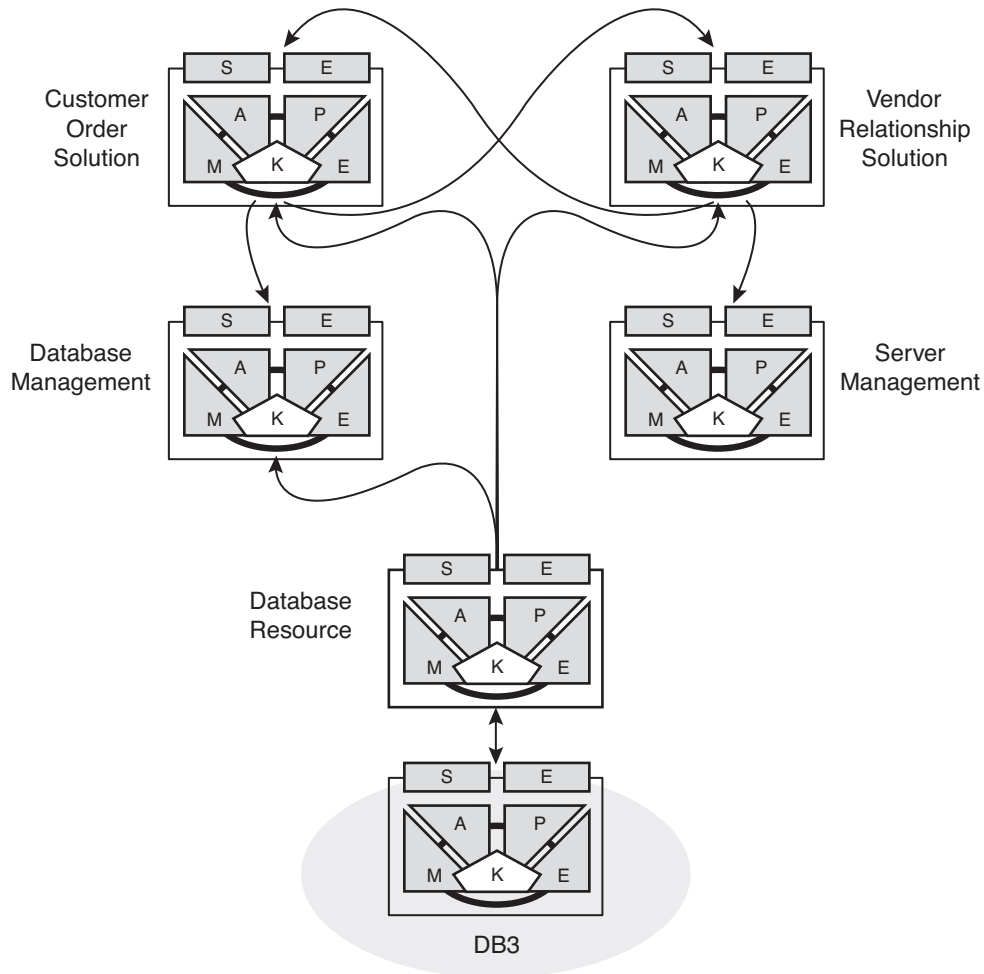
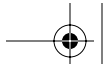


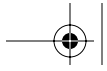
FIGURE 3.14 How six autonomic managers directly and indirectly manage the DB 3 resource.

Autonomic Manager Knowledge

Data used by the autonomic manager’s four components is stored as shared knowledge. The shared knowledge includes things like topology information, system logs, performance metrics, and policies.

The knowledge used by a particular autonomic manager could be created by the monitor part, based on the information collected through sensors, or passed into the autonomic manager through its effectors. An example of the former occurs when the monitor part creates knowledge based on recent





activities by logging the notification it receives from a managed element into a system log. An example of the latter is a policy. A policy consists of a set of behavioral constraints or preferences that influences the decisions made by an autonomic manager. Specifically, the plan part of an autonomic manager is responsible for interpreting and translating policy details. The analysis part is responsible for determining if the autonomic manager can abide by the policy, now and in the future.

Self-Managing Systems Change the IT Business

Ideally, the IT business operates through a collection of best practices and processes. Principles of the IT Infrastructure Library (from the Office of Government Commerce in the UK) and the IBM IT Process Model (developed by IBM Global Services) influence key IT best practices and processes. Figure 3.15 shows an example of a typical process flow for incident management, problem management, and change management. The actual mechanics of how these flows are implemented in a particular IT organization varies, but the functionality remains the same.



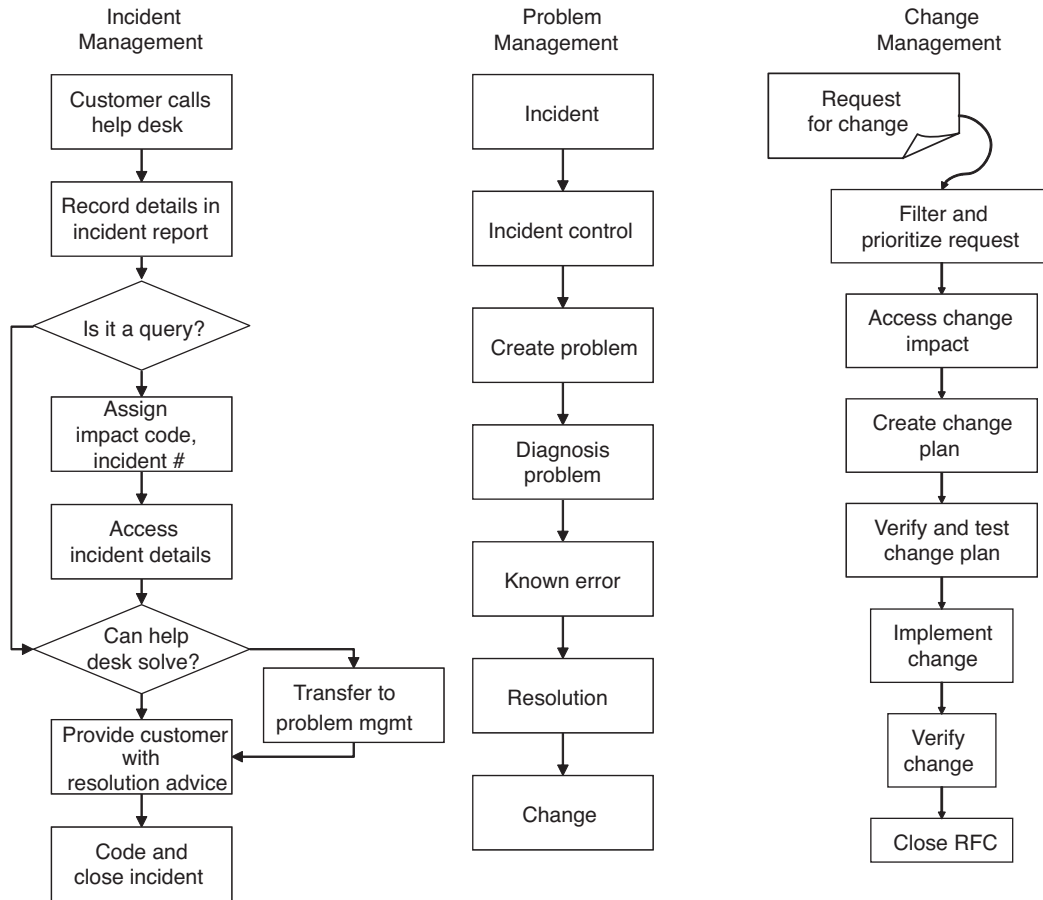
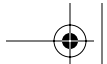


FIGURE 3.15 How typical IT processes can be represented as autonomic control loops.

The efficiency and effectiveness of these processes are measured using metrics such as elapsed time to complete a process, percentage executed correctly, and people and material costs to execute a process. Autonomic systems can positively affect these types of metrics, improving responsiveness, reducing TCO, and enhancing TTL through:

- *Quick process initiation*—Typically, implementing these processes requires an IT professional to initiate the process, create the request for change, spend time collecting incident details, and open a problem record. In a self-managing system, components can initiate the processes based on information derived directly from the system. This helps reduce the manual labor and time required to respond to critical



situations, resulting in two immediate benefits: more timely initiation of the process and more accurate data from the system.

- *Reduced time and skill requirements*—Some tasks or activities in these processes usually stand out as skills-intensive, long-lasting, and difficult to complete correctly because of system complexity. In a change management process, such an activity is the “change impact analysis task.” In problem management, such an activity is problem diagnosis. In self-managing systems, resources are built so that the expertise required to perform these tasks can be encoded or automated into the system. This helps reduce the amount of time and degree of skill needed to perform these tedious tasks, since technology rather than people can perform the tasks.

The mechanics and details of IT processes, such as change management and problem management, are different, but it is possible to categorize these into four common functions: Collect the details, analyze the details, create a plan of action, and execute the plan. These four functions correspond to the monitor, analyze, plan, and execute parts of the architecture. The approximate relationship between the activities in some IT processes and the parts of the autonomic manager are illustrated in Figure 3.16:



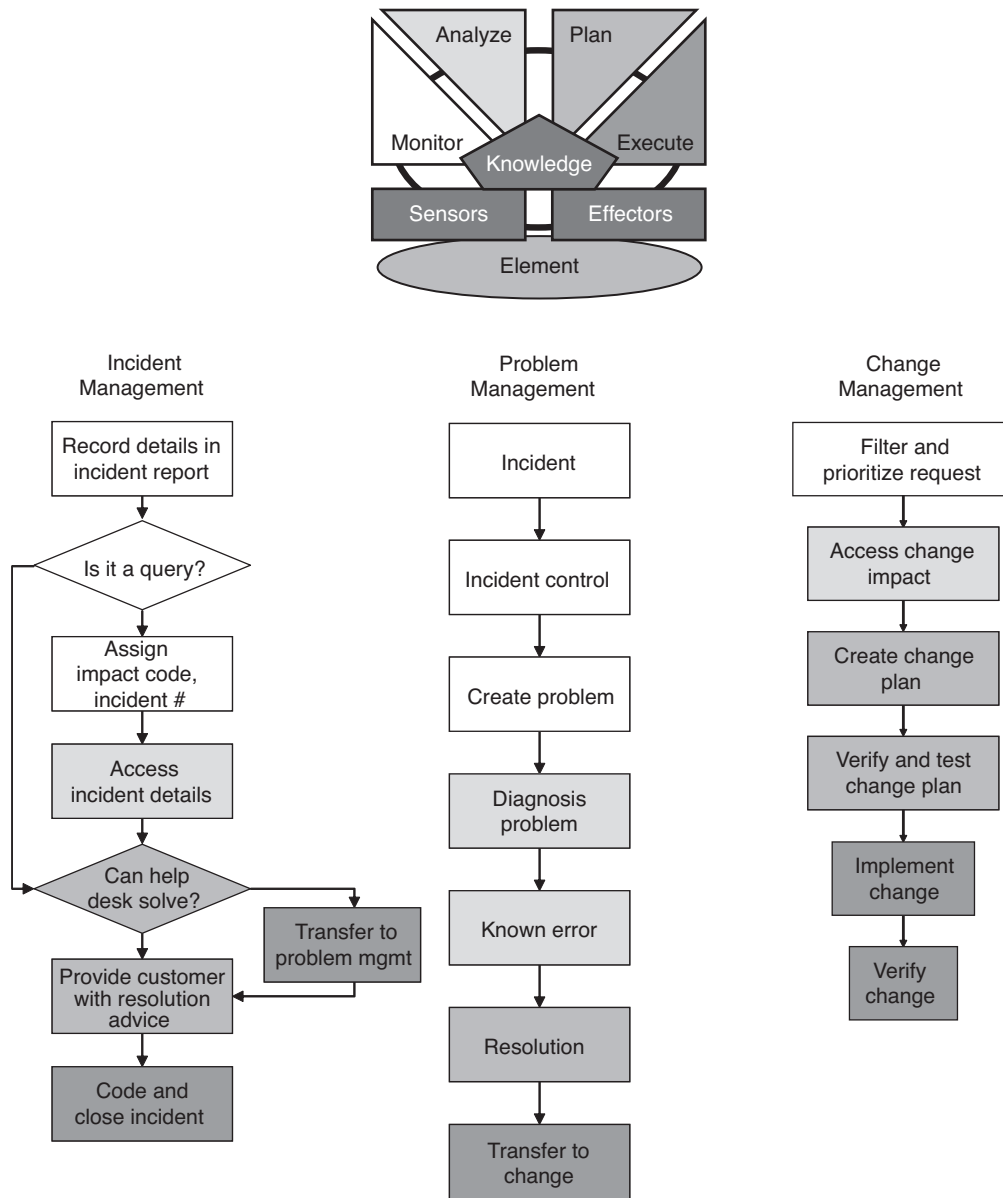


FIGURE 3.16 How Autonomic Computing affects IT processes.

The analyze and plan mechanisms are the essence of an Autonomic Computing system because they encode the “know-how” to help reduce the skill and time required of the IT professional.



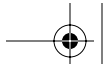
An Evolution, Not a Revolution: Levels of Management Maturity and Sophistication

Incorporating autonomic capabilities into a computing environment is an evolutionary process enabled by technology. It is ultimately implemented by an enterprise through the adoption of these technologies, supporting processes, and skills. Throughout the evolution, the industry will continue delivering self-management tools to improve IT professionals' productivity.

To understand the level of sophistication of the tools and capabilities that are—and will be—delivered by the industry, consider the following five levels of autonomic maturity, which are also illustrated in Figure 3.17:

- At the *basic level*, IT professionals manage each infrastructure element independently and set it up, monitor it, and eventually replace it.
- At the *managed level*, systems management technologies can be used to collect information from disparate systems onto fewer consoles, helping to reduce the time it takes for the administrator to collect and synthesize information as the IT environment becomes more complex.
- At the *predictive level*, new technologies are introduced to provide correlation among several infrastructure elements. These elements can begin to recognize patterns, predict the optimal configuration, and provide advice on what course of action the administrator should take.
- At the *adaptive level*, the IT environment can automatically take action based on the available information and knowledge of what is happening in the environment. As these technologies improve and as people become more comfortable with the advice and predictive power of these systems, the technologies can progress to the autonomic level.
- At the *autonomic level*, business policies and objectives govern the IT infrastructure operation. Users interact with the autonomic technology tools to monitor business processes, alter the objectives, or both.





	Basic Level 1	Managed Level 2	Predictive Level 3	Adaptive Level 4	Autonomic Level 5
Characteristics	Rely on system reports, product documentation, and manual actions to configure, optimize, heal and protect individual IT components	Management software in place to provide consolidation, facilitation and automation of IT tasks	Individual IT components and systems able to monitor, correlate and analyze the environment and recommend actions	IT components, individually and collectively, able to monitor, correlate, analyze and take action with minimal human intervention	Integrated IT components are collectively and dynamically managed by business rules and policies
Skills	Requires extensive, highly skilled IT staff	IT staff analyzes and takes actions	IT staff approves and initiates actions	IT staff manages performance against SLAs	IT staff focuses on enabling business needs
Benefits	Basic requirements addressed	Greater system awareness Improved productivity	Reduced dependency on deep skills Faster/better decision making	Balanced human/system interaction IT agility and resiliency	Business policy drives IT management Business agility and resiliency
Manual					Autonomic

FIGURE 3.17 The Autonomic Computing evolution occurs gradually across five phases.

How Technology Must Evolve to Support Autonomic Computing

The earlier discussion about autonomic maturity levels demonstrated that self-managing capabilities would not be incorporated in one quick step. Rather, they constitute a concept that permeates all aspects of a system. Figure 3.18 reinforces this observation by showing a possible relationship between the maturity levels, the three decision-making contexts (resource element context, composite resource context, and business solution context), and the parts of the autonomic manager. This mapping results in two important observations:

- First, as the maturity levels increase, the decision-making context for the autonomic manager changes. The pyramid on the right-hand side summarizes the three different decision-making contexts in which autonomic managers can implement self-managing capabilities.
- Second, different parts of the autonomic manager are implemented at each maturity level. The monitor and execute parts of the autonomic manager are implemented at the basic and managed levels. So, at these two levels, IT professionals are responsible for performing the func-



tions of the analyze and plan parts. The analyze part of the autonomic manager is supplied at the predictive maturity level. At this level, the IT professional is responsible for the plan function. At the adaptive and autonomic levels, all the parts of the autonomic manager are working, so the IT professional can delegate work to the system. The difference between these two maturity levels is the decision-making context. The adaptive maturity level supports either the resource element or the composite element context, and the autonomic level supports the business solution context.

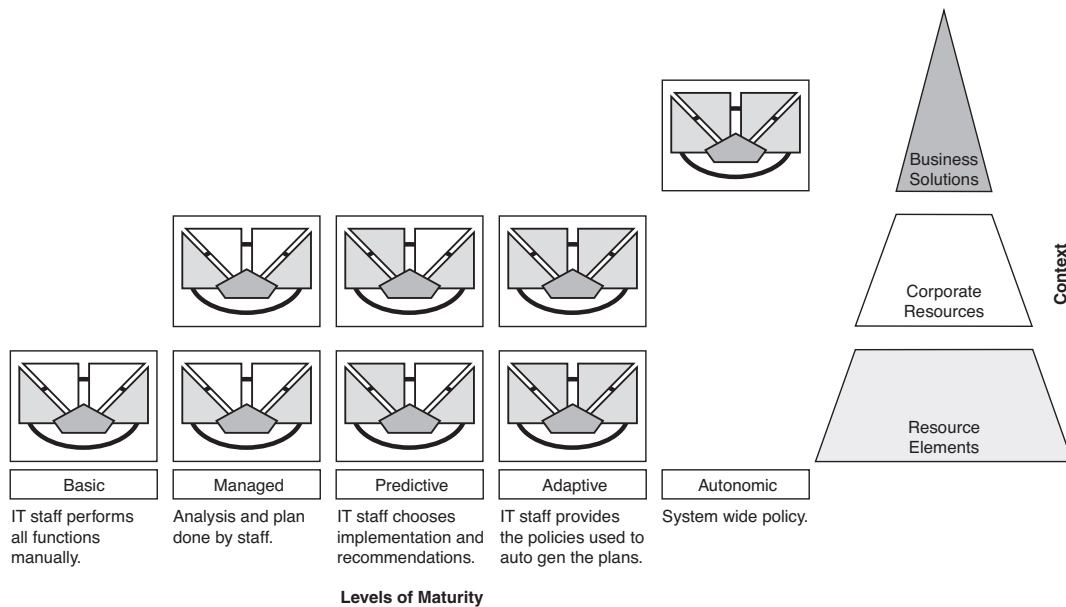
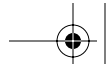


FIGURE 3.18 By progressing along the five autonomic maturity levels, businesses can evolve IT environments to fully autonomic levels.

As Figure 3.18 shows, the progressive implementation of the architecture occurs for each of the three contexts. This is because it is difficult to deliver a self-managing capability in a business system context if there is no self-managing capability in the lower contexts.

Core Autonomic Capabilities

For the autonomic managers and managed elements in an autonomic system to work together, the developers of these components need a common



set of capabilities. This section describes an initial set of core capabilities that are needed to build autonomic managers. These core capabilities include: solution knowledge, common system administration, problem determination, autonomic monitoring, complex analysis, policy for autonomic managers, and transaction measurements. Technologies that deliver these capabilities will accelerate the delivery of autonomic managers that can collaborate in an autonomic system.

Solution Knowledge

Today, there are a myriad of installation, configuration, and maintenance mechanisms. The differences and idiosyncrasies of these many system administration tools and distribution packaging formats create significant problems in managing complex system environments. These problems are further compounded in a Web services environment, where application functionality can be composed dynamically. From an autonomic systems perspective, lack of solution knowledge inhibits important elements of self-configuring, self-healing, and self-optimizing.

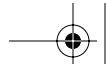
A common solution knowledge capability eliminates the complexity introduced by many formats and installation tools. By capturing installation and configuration information in a consistent manner, it creates knowledge that autonomic managers can use in contexts beyond installation, such as problem determination or optimization. Solutions are combinations of platform capabilities (operating systems and middleware) and application elements (such as Enterprise JavaBeans [EJB], DB2 tables, hypertext markup language [HTML] pages, and flow definitions) that solve a particular customer problem.

The Autonomic Computing Blueprint defines a set of constructs for composing installable units and design patterns that make it possible to standardize solution knowledge. An *installable unit* is composed of a descriptor that describes the content of the installable unit and the actual artifact to be installed. The descriptor and artifact comprise the package—like a Java archive file. The target environment for the installable unit is called the *hosting environment*, or the container that will accept the artifact to be installed.

There are three categories of installable units that build on each other:

- *Smallest installable unit*—This unit contains one atomic artifact.
- *Container installable unit*—This unit aggregates a set of artifacts for a particular container type.
- *Solution module installable unit*—This unit contains multiple instances of container installable units.





The Autonomic Computing Blueprint identifies a number of enabler technology components for solution knowledge. These include:

- *A dependency checker*—This determines whether the dependency of an artifact is satisfied in the targeted hosting environment.
- *An installer*—The functionality that knows how to extract the artifacts in the installable units and invoke the appropriate operations on the target hosting environment.
- *An installable unit database*—A library for installable units.
- *Deploy logic*—This functionality knows how to distribute an installable unit to an installer component.
- *An installed unit “instances” database*—This database stores the configuration details about installable units and hosting environments.

The installable unit schema definitions and enabler components create the basis for coherent installation, configuration, and maintenance processes at the solution level versus different product-specific mechanisms.

Common System Administration

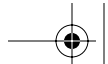
Autonomic systems require common console technology to create a consistent human-facing interface for the autonomic manager elements of the IT infrastructure. The common console capability provides a framework for reuse and consistent presentation for other autonomic core technologies.

The primary goal of a common console is to provide a single platform that can host all the administrative console functions in server, software, and storage products in a manner that allows users to manage solutions rather than managing individual systems or products. Administrative console functions range from setup and configuration to solution runtime monitoring and control.

The values to the customer in having a common administrative console are: reduced cost of ownership, attributable to more efficient administration, and reduced learning curves as new products and solutions are added to the autonomic system environment. The reduced learning curve results from using both standards and the familiar Web-based presentation style. By enabling increased consistency of presentation and behavior across administrative functions, the common console creates a familiar user interface that promotes reusing learned interaction skills versus learning new, different, product-unique interfaces.

The common console functionality could be a platform for IBM products with extensions for ISVs and business partners. Common console interfaces





could also be made available outside IBM to enable development of new components for IBM products or to enable bundling of common console components in non-IBM products. Since the common console architecture is standards-based, IBM could propose it as a system administration user interface infrastructure standard.

A common console instance consists of a framework and a set of console-specific components provided by other product development groups. Administrative activities are executed as portlets. Consistency of presentation and behavior is key to improving Autonomic Computing system administrative efficiency, and will require ongoing effort and cooperation among many product communities. Console guidelines will take time to emerge, given the large number of human factors and design organizations involved.

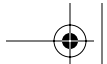
Problem Determination

Whether healing, optimizing, configuring or protecting, autonomic managers take actions based on problems or situations they observe in their managed elements. Therefore, one of the most basic capabilities is being able to extract high-quality data to determine whether or not a problem exists in a managed element. In this context, a problem is a situation in which an autonomic manager needs to take action. A major cause of poor-quality information is the diversity in the format and content of the information provided by the managed element.

There is a relatively small, finite, canonical set of situations that is reported by components. This common set covers a large percentage of the situations that are reported by most system components. Currently, components use different terminology to report common situations. For example, one component may report the situation that a “component has started,” where another component may report that the “component has begun execution.” This variability in the description of the situation makes writing and maintaining autonomic systems difficult.

To address this diversity of the data collected, the Autonomic Computing Blueprint requires a common problem determination architecture that normalizes the data collected, in terms of format, content, organization, and sufficiency. To do this, it defines a base set of data that must be collected or created when a situation or event occurs. This definition includes information on both the kinds of data that must be collected as well as the format that must be used for each field collected. The problem determination architecture categorizes the collected data into a set of situations, such as component starts and stops.





The technologies used to collect autonomic data must be capable of accommodating legacy data sources (e.g., logs and traces) as well as data that is supplied using the standard format and categorization. To accommodate this legacy data, the architecture defines an adapter/agent infrastructure that provides the ability to plug in adapters to transform data from a component-specific format to the standard format as well as sensors to control data collection (e.g., filtering, aggregation, etc.).

Autonomic Monitoring

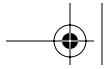
Autonomic monitoring is a capability that provides an extensible runtime environment for an autonomic manager to gather and filter data obtained through sensors. Autonomic managers can utilize this capability as a mechanism for representing, filtering, aggregating, and performing analyses on sensor data. This capability includes:

- A common way to capture the information that surfaces from managed elements through sensors. This should utilize the CIM, SNMP, Windows Management Instrumentation (WMI), and JMX industry standards.
- Built-in sensor data-filtering functions.
- A set of pre-defined resource models (and a mechanism for creating new models) that enables the combination of different pieces of sensor data to describe the state of a logical resource. Resource models describe business-relevant “logical objects” from the perspective of common problems that can affect those objects. Examples of frequently used resource models include “machine memory” and “machine connectivity.”
- A way to incorporate policy knowledge.
- A way to plug in analysis engines that can provide basic event isolation, basic root cause analysis, and server-level correlation across multiple IT systems, and automate initiation of corrective actions.

An autonomic manager using this autonomic monitoring functionality can help manage certain applications or resources more effectively through:

- *Multiple source data capture*—Allows processing of data from industry-standard APIs, and from any custom data interfaces that a particular application uses
- *Local persistence checking*—Links corrective actions or responses to the repeated occurrence of a problem condition so that a single point-in-time threshold exception does not immediately trigger a costly and unnecessary troubleshooting response





- *Local intelligent correlation*—Recognizes a number of metrics in aggregate as a “problem signature,” enabling root cause identification and responses to problems rather than symptoms
- *Local data store and reporting*—Provides a real-time “heart monitor” that determines whether the application environment and individual applications are functioning properly

The reference model component of autonomic monitoring should provide built-in intelligence, a set of embedded best-practices data that:

- Interprets the quality of a logical object against a defined baseline
- Logs performance data related to the business object
- Proactively manages the application through a pre-defined collection of problem signatures

This resource management model demonstrates a plurality of capabilities, and can be exploited in a number of ways to:

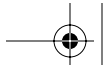
- Manage systems and resources reactively, allowing escalation of the status change of a resource
- Adopt a proactive management strategy, using resource models to automatically diagnose and fix problems at the local level
- Use predictive management tasks, allowing the utilization of data mining tools to analyze performance metrics and predict abnormal behavior
- Use adaptive management, automatically tuning model baselines based on historical trends of performance data collected by the model itself

Complex Analysis

Autonomic managers need to have the capability to perform complex data analysis and reasoning on the information provided through sensors. The analysis will be influenced by stored knowledge data. The Autonomic Computing Blueprint defines complex analysis technology building blocks that autonomic managers can use to represent knowledge, perform analysis, and do planning.

Complex analysis technology components and tools provide the power and flexibility required to build practical autonomic managers. Autonomic managers must collect and process large amounts of data from sensors and managed resources. This data includes information about resource configuration, status, offered workload, and throughput. Some of the data is static or changes slowly, while other data is dynamic, changing continuously through time. An autonomic manager’s ability to quickly analyze and





make sense of this data is crucial to its successful operation. Common data analysis tasks include classification, clustering of data to characterize complex states and detect similar situations, prediction of anticipated workload and throughput based on past experience, and reasoning for causal analysis, problem determination, and optimization of resource configurations.

The complex analysis technology uses a rule-based language that supports reasoning through procedural and declarative rule-based processing of managed resource data. The underlying rule engines can be used to analyze data with scripting as well as forward and backward inference methods using if-then rules, predicates, and fuzzy logic. Application classes can be imported directly into rule sets so that data can be accessed (using sensors) and control actions can be invoked directly from rules (using effectors). The rule-based language features both Java programming language-like text and XML source rule set representations, enhancing productivity for rule authors familiar with Java syntax and allowing portable knowledge interchange. Rule sets can include multiple rule blocks so that a mix of procedural and inference methods can be used to analyze data and define autonomic manager behavior.

Other complex analysis technology components can be used to augment the basic rule capabilities. These include Java Beans to access data from flat files and relational databases, to filter, transform, and scale data using templates, and to write data to flat files and databases. Complex analysis technology also includes machine learning beans and agents to perform classification, clustering, and time-series prediction using neural networks, decision trees, and Bayesian classifiers, and to perform statistical data analysis and optimization using genetic algorithms.

As a core Autonomic Computing technology, complex analysis components can enhance productivity when used as building blocks to implement specific plan, analyze, or knowledge functionality.

Policy for Autonomic Managers

An Autonomic Computing system requires a uniform method for defining the policies that govern the decision-making for autonomic managers. A *policy* specifies the criteria that an autonomic manager uses to accomplish a definite goal or course of action. As shown in Figure 3.19, policies are a key part of the knowledge used by autonomic managers to make decisions, essentially controlling the planning portion of the autonomic manager. By defining policies in a standard way, they can be shared across autonomic managers to enable entire systems to be managed by a common set of policies.



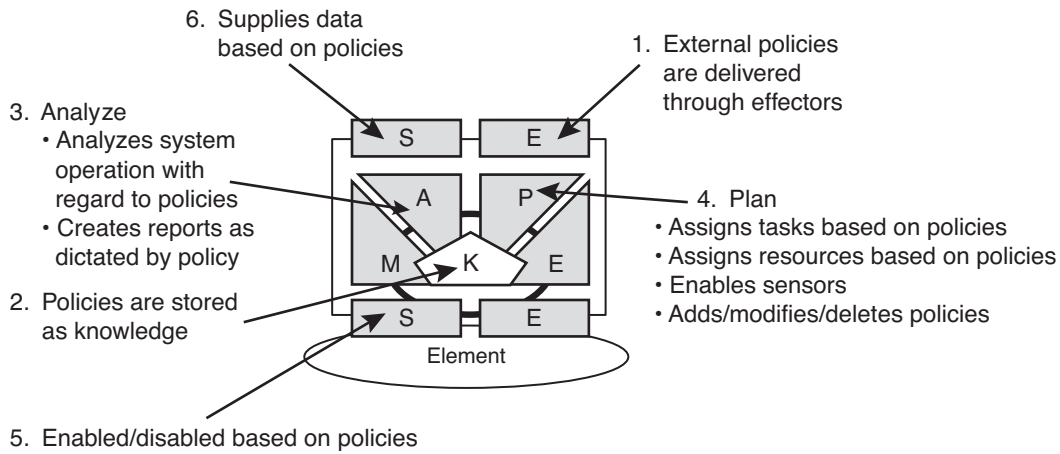
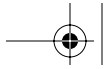


FIGURE 3.19 Policy-based management in autonomic managers.

Today, the term “policy” is used in various contexts to mean seemingly different things, and exists in various forms and formats. Table 3.1 contains some examples:

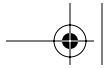
TABLE 3.1 Forms of Autonomic Policies

Typical Domain	Examples
IT resource policies	If a packet is gold, queue it with high priority.
Business process policies	If it's a frequent customer, apply a 3% discount.
Interaction policies	Require Kerberos authentication.
SLA policies	If 2-second response time is not delivered, refund 30%.

Despite the apparent differences, these examples exhibit substantial commonality and must be specified consistently for an autonomic system to behave cohesively. The Autonomic Computing Blueprint is currently defining the specifications and capabilities for policy-based autonomic managers. This definition includes:

- Specification of canonical configuration parameters for management elements
- Format and schema to specify user requirements or criteria
- Mechanisms, including wire formats, for sharing and distributing policies
- Schema to specify and share policy among autonomic managers





One of the key functions that an autonomic system must perform is to share policies among autonomic managers, so this capability will leverage and extend policy standards.

Transaction Measurements

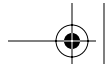
Autonomic managers need a transaction measurement capability that spans system boundaries to understand how the resources of heterogeneous systems combine into a distributed transaction execution environment. By monitoring these measurements, the autonomic manager can analyze and plan to change resource allocations to optimize performance across these multiple systems according to policies, as well as determine potential bottlenecks in the system.

Tuning servers individually cannot ensure the overall performance of applications that span a mix of platforms. Systems that appear to be functioning well on their own may not, in fact, be contributing to optimal end-to-end processing. Inefficiencies created by infrastructure complexity and the growing number of servers are outstripping the increasing productivity provided to system administrators by powerful management tools. While hardware and software costs decline, people costs rise and larger staffs are needed. Additionally, when hundreds or even thousands of different servers are involved, it may not be possible with current technology for any number of administrators to discover failing systems in time to isolate or repair them before any damage is done.

Furthermore, the average utilization of most distributed systems is very low today. Many on demand business applications must be capable of handling large spikes in volume, so companies typically buy hardware to meet the needs of those spikes. However, when the original application is not fully utilizing computing resources, there is no easy way to divert the excess capacity to lower priority work. Therefore, customers must buy and maintain a separate infrastructure for each application to meet that application's most demanding computing needs.

Instituting an end-to-end transaction measurement infrastructure enables a distributed workload management capability, and addresses these problems of rising administrative costs and low hardware utilization. The general philosophy behind distributed workload management is one of policy-based, goal-oriented management. The philosophy requires both a policy definition infrastructure (like that mentioned above) and an end-to-end transaction measurement infrastructure. The policy contains simple definitions of classes of service—broad categories of “work”—and an associated perfor-





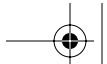
mance goal for each class of service. The goals are stated in terms such as “need to complete 90 percent in less than 1 second,” or “an average response time of 2 seconds.” In addition to a goal declaration, each class of service is accompanied by a business importance level, which indicates how important the achievement of the goal is to the business that owns the computing resources. The relationships are then quite simple: satisfy the goals of the most important workloads and then worry about the rest.

Once the service classes are defined and prioritized, the next step is to understand what systems are used to process the service classes, and to instrument these systems appropriately. An administrator already understands what servers are in place, and what applications are on each; but what the administrator typically does not know is the exact nature of the relationships between application environments, and the relationships between the various servers. Therefore, by applying instrumentation across the application environments uniformly, an administrator can determine these relationships and the flow of transactions through the system. The application response measurement (ARM) API is key to providing this uniform instrumentation.

The key capability needed for a distributed workload management system is the ability to understand the transaction topology and map the service classes to this topology. Autonomic managers involved in workload management need a transaction measurement capability to understand how the systems involved commit their resources to execute the workload, and how changes in allocation affect performance over time.

The combination of prioritized service classes and an understanding of the systems involved in delivering a particular class of service enables distributed workload management. This creates a general workload management infrastructure that can be used for many purposes because of the ability to prioritize different service classes according to the needs of the business. The distributed workload management system can optimize work across the distributed infrastructure in an attempt to meet all the goals associated with each service class. If all the goals cannot be achieved, then changes can be made to ensure that the most important applications meet their goals first. Overall, this enables a single infrastructure to, in an autonomic manner, “self-optimize” while meeting the needs of the business.





Standards for Autonomic Computing

The fundamental nature of Autonomic Computing systems precludes any one company from delivering a total autonomic solution. Enterprises have heterogeneous IT infrastructures and must deal with heterogeneous environments outside the enterprise. A proprietary implementation would be like a heart that maintains a regular, steady heartbeat but is not able to adjust to the needs of the rest of the body when under stress.

Autonomic Computing systems require deployment of autonomic managers throughout the IT infrastructure, managing resources that include other autonomic managers from a diverse range of suppliers. These systems, therefore, must be based on open industry standards.

The Autonomic Computing Blueprint identifies relevant existing computing industry standards. New open standards will continue to be developed and shared with the industry that will define new mechanisms for interoperating in a heterogeneous system environment.

Summary

Self-management is about shifting the burden of managing systems from people to technologies. When the self-management capabilities delivered by IBM and other vendors are able to collaborate, it will be possible to deliver Autonomic Computing capabilities for the entire IT infrastructure. In these environments, the elements of a complex IT system will manage themselves based on a shared view of system-wide policy and objectives.

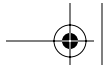
FUTURE OF COMPUTING

IBM has named its vision for the future of computing “Autonomic Computing.” This new paradigm shifts the fundamental definition of the technology age from one of computing, to one defined by data. Access to data from multiple, distributed sources, in addition to traditional centralized storage devices, will allow users to transparently access information when and where they need it.

At the same time, this new view of computing will necessitate changing the industry’s focus on processing speed and storage to one of developing distributed networks that are largely self-managing, self-diagnostic, and transparent to the user.

This chapter presented a high-level blueprint to assist in delivering Autonomic Computing in phases. The architecture reinforces the fact that self-managing implies intelligent control loop implementations that will execute in one of three decision-making contexts to monitor, analyze, plan, and exe-





cute using knowledge of the environment. In addition, the loops can be embedded in resource runtime environments or delivered in management tools. These control loops collaborate using a matrix management protocol.

The journey to a fully autonomic IT infrastructure is an evolution. The stages of this evolution were illustrated by showing which aspects of the architecture need to be addressed at the five management maturity levels. This model was then applied to the IT infrastructure using the three decision-making contexts.

Enterprises want and need to reduce their IT costs, simplify the management of their IT resources, realize a faster return on their IT investment, and ensure the highest possible levels of system availability, performance, security, and asset utilization. Autonomic Computing addresses these issues—not just through new technology, but also through a fundamental, evolutionary shift in the way IT systems are managed. Moreover, such systems will free the IT staff from detailed, mundane tasks, allowing them to focus on managing their business processes. True Autonomic Computing will be accomplished through a combination of process changes, skills evolution, new technologies, architecture, and open industry standards.

For readers wishing to further their understanding of the IBM Corporation's Autonomic Computing initiative, please refer to the book entitled *Autonomic Computing* by Richard Murch.

In the next chapter, we will explore the complementary side to Autonomic Computing, which together with autonomic disciplines enables very powerful enterprise-wide solutions for integrating into on demand business solutions. This second, complementary computing discipline is Grid Computing.

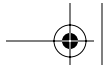
Glossary of Autonomic Computing Terms

This section contains definitions of some Autonomic Computing terms that were utilized in this chapter.

Analyze: The function of an autonomic manager that models complex situations.

Autonomic: Being accomplished without overt thought or action. Example: the human autonomic nervous system that monitors and regulates temperature, pupil dilation, respiration, heart rate, digestion, etc.





Autonomic Computing: An approach to self-managed computing systems with a minimum of human interference.

Autonomic manager: A part of an autonomic element that manages a managed element within the same autonomic element.

Data collection: Definitions for standard situational event formats in the Autonomic Computing architecture (also called logging). This notion is one of the Autonomic Computing core technologies.

Domain: A collection of resources that have been explicitly or implicitly grouped together for management purposes.

Effector: A way to change the state of a managed element.

Execute: The function of an autonomic manager that is responsible for interpreting plans and interacting with element effectors to insure that the appropriate actions occur.

Install (Installation): Definitions for standard methods to describe software deployment and installation. This notion is one of the Autonomic Computing core technologies.

Knowledge: The common information that the monitor, analyze, plan, and execute functions require to work in a coordinated manner.

Maturity index: A graduated scale that expresses the level of maturity of Autonomic Computing, where Level 1 is basic (completely manual), Level 2 is managed, Level 3 is predictive, Level 4 is adaptive, and Level 5 is completely autonomic.

Open Grid Services Architecture (OGSA): A Grid Computing system architecture based on an integration of Grid Computing and Web services concepts and technologies.

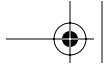
Plan: The function of an autonomic manager that provides a way to coordinate interrelated actions over time.

Policy: A definite goal, course, or method of action to guide and determine future decisions. Policies are implemented or executed within a particular context. This is a set of behavioral constraints and preferences that influence decisions made by an autonomic manager. This notion of policy utilization is one of the Autonomic Computing core technologies.

Policy-based management: A method of managing system behavior or resources by setting policies that the system interprets.

Self-configuring: Setting an element up for operation.





Self-healing: Repairing damage to an element regarding its own operational integrity.

Self-managing: Directing and controlling an element. This is most often regarding self-configuring, self-optimizing, self-protecting, and self-healing operations.

Self-optimizing: Tuning or improving an element's own performance.

Self-protecting: Maintaining an element's own operational integrity.

Sensor: A way to get information about a managed element.

Situations: Events that Autonomic Computing components report to the outside world. Situations vary in granularity and complexity, ranging from simple situations like the start of a component to more complex situations like the failure of a disk subsystem.

