Toward an on demand service-oriented architecture

C. H. Crawford
G. P. Bate
L. Cherbakov
K. Holley
C. Tsocanos

The success of an on demand e-business requires that business process, application, and information technology (IT) infrastructure integration merge into a comprehensive and cohesive architecture, where business process transformation drives service-oriented development and on demand enterprise computing. This enabling architecture is often described as a service-oriented architecture (SOA) and is a prerequisite accelerator for on demand solutions. The primary focus of SOA has been on dynamic reconfiguration of services from defined business processes, and on developing business services based on Web services and, more recently, grid services. Current descriptions of SOA are less focused on overall IT infrastructure enablement, both from a business policy perspective and within the context of service-oriented development. In this paper, we extend the current thinking on SOA to include a more comprehensive integration of business process transformation and the enabling technologies of service-oriented development and policy-based IT management. We call this extension *on demand SOA.* We develop these concepts by using an existing scenario: a financial services sector "Life Change" business process scenario, which involves distributed and disjoint transactions as well as stateless high-performance computing (HPC) applications.

Over the last 40 years, information technology (IT) architectures and development approaches have dealt with increasing levels of IT complexity and integration challenges. Constrained budgets continue to mandate that legacy systems be reused rather than replaced. Growth by merger and acquisition requires that entire IT organizations be integrated and absorbed. Additionally, easy access to the Web has created the possibility for new business models, which must be evaluated for their potential.

At the same time, the traditional needs of IT organizations persist—primarily focused on quick response to new requirements, typically consisting in turn of corporate management pushing for better IT utilization, skills simplification, greater return on investment (ROI), continued integration of historically separate systems, and faster implementation of new ones. The endless varieties of hardware, operating systems, middleware, languages, and data

storage create an environment where accommodation of heterogeneity is fundamental to system development. The cumulative effects of decades of growth and evolution, encompassing multiple computing architectures, programming languages, and connectivity products, have produced the complexity that now challenges all IT organizations.

As the level of complexity continues to increase, traditional approaches are reaching their limits. Powerful new technologies such as Web services,[1] autonomic computing,[2] utility computing,[3] and grid services[4] provide partial answers, but the problem in many cases is the lack of a consistent architecture within which applications can be rapidly developed, integrated, and reused. SOA (service-oriented architecture) holds the promise of being the consistent architecture required for future development. In isolation, however, neither the application of these new technologies nor the use of the SOA approach, itself, provides a complete solution.

ON DEMAND SOA

Much has been written about why SOA, Web services, autonomic computing, utility computing, and grid technologies and standards can be beneficial, but a holistic view, unconstrained by technology, is currently lacking. *On demand SOA* provides such a view, namely a distributed computing model infused with the building blocks of these new technologies. Taken separately, each provides significant benefits, but the integration of these technologies promises far greater impact and provides the foundation for on demand SOA.

To demonstrate on demand SOA, we use a real-life example of a business-to-business (B2B) process in which services are used to automate the processing of an electronic purchase order request (POR). Our buyer, Acme, Inc., is a large manufacturing company. Our supplier, Pens R Us, is a large stationery company. A contract already exists between the two parties. Acme, Inc. wants to use an electronic POR to buy 500 reams of paper from Pens R Us. The first step is to determine which services are required to fulfill the process. At one extreme POR may be viewed as a single service; at the other extreme the service granularity could be so fine that the POR might be constructed from multiple services. The choice is made by balancing quality of service (QoS) characteristics, ubiquitous service reuse, and reduction of complexity for service composition.

The supplier may choose to view the process as the following steps:

1. Supplier authenticates the purchaser.
2. Supplier looks up the buyer contract based on purchaser ID.
3. Purchaser browses the product catalog with negotiated prices from the contract.
4. Purchaser adds items to the shopping cart.
5. Purchaser checks out, providing payment description and delivery information.
6. Order information is sent to the fulfillment department.
7. Confirmation of order with expected delivery date is sent to the purchaser.

From such a process description, we can further list the software or application services that are required:

1. Login
2. Contract lookup
3. Catalog browsing with shopping cart and checkout
4. POR data creation (from login ID, contract ID, shopping cart data, and other information supplied by checkout)
5. Information delivery to fulfillment process
6. Message to purchaser to confirm order

The enablement of SOA with open standards, for example, Simple Object Access Protocol (SOAP), Extensible Markup Language (XML), Web Services Description Language (WSDL), and Open Grid Services Architecture (OGSA), among others, offers the ability to fulfill the promises and value propositions of SOA implementations. These open standards allow a service to be decoupled from the IT infrastructure. As long as vendor support for the standards exists across resources, the service composer need not be concerned with where the service will run—only with how to assemble flows between services. Additionally, dynamic service lookup means that service consumers need not be concerned with where underlying software resources exist on distributed, heterogeneous systems.

To achieve the promised benefits of SOA, one needs all of these technologies working together in a meaningful way to create what we define as on demand SOA. John Hagel has asserted that, "Over time, distributed service architectures enabled by

Web services technologies have the potential to become the dominant technology architecture for all business activities."[5] When we include autonomic computing, grid services, and Web services, this vision becomes a reality, and we have on demand SOA.

Benefits of on demand SOA

The key benefit of on demand SOA is allowing business processes to be responsive with a high degree of flexibility. In turn, for business processes to be responsive, a high degree of automation and adaptability must be enabled with the underlying IT application and technical architectures. Organizations that adopt an on demand SOA will realize several benefits:

■ The key benefit of on demand SOA is allowing business processes to be responsive with a high degree of flexibility ■

- *Leveraging of existing assets*—Using a suitable SOA framework, a business service can be constructed as an aggregation of existing components and made available to the enterprise. Using this new service only requires knowing its interface and name. The service's internal details are hidden from the outside world, as is the complexity of the data flow through the components that make up the service. This component anonymity allows organizations to leverage current investments, constructing services from a conglomeration of components built on different machines, running different operating systems, and developed in different programming languages. Legacy systems can be encapsulated and accessed through Web services interfaces.

- *Commoditization of infrastructure*—Infrastructure development and deployment will become more consistent across different enterprise applications. Existing components, newly developed components, and components purchased from vendors can be consolidated within a well-defined SOA framework. Such aggregations of components will be deployed as services on the existing infrastructure, resulting in the underlying infrastructure becoming essentially a commodity element.

- *Faster time to market*—Line-of-business (LOB) and enterprise-Web-services libraries will become core assets for enterprises adopting the SOA framework. Building and deploying services with these Web services libraries will reduce time to market dramatically as new initiatives reuse existing services and components, thus reducing design, development, testing, and deployment time.

- *Reduced cost*—As business demands evolve and new requirements are introduced, the cost of enhancing and creating new services is greatly reduced by adopting the SOA framework and the services library for both existing and new applications. The IT infrastructure can be further optimized by using grid services or utility computing.

- *Risk mitigation*—Reusing existing components reduces the risk of introducing new errors, and thus potential points of failure, into the process of enhancing or creating new business services.

- *Continuous business process improvement*—SOA allows a clear representation of process flows identified by the order of the services used in a particular business process. This provides business users with an ideal environment to monitor business operations. Process modeling is reflected in the business service. Process manipulation is achieved by reorganizing the pieces in a pattern (the components that constitute a business service). This allows users to change process flows while monitoring the resulting effects, thus facilitating continuous improvement.

- *Customer-centric architecture*—Existing architecture models and practices tend to be application-centric. As organizations evolve from an application orientation toward a service orientation, consumers or customers of the applications benefit. A customer-centric architecture provides a user-configurable approach versus a one-size-fits-all or prepackaged workflow. This enhanced flexibility is brought about by identifying and exposing services, as opposed to locking users into hardwired application flows.

On demand SOA requirements

Maximum benefit from an SOA implementation in an on demand environment occurs when the underlying enterprise computing infrastructure is virtualized and enabled for policy-based system management and when the IT policies have been derived in turn from the corresponding business policies.

The current focus in SOA is on process definition and application enablement by means of Web services or grid technologies and standards. The link between business process policy and IT policy through application enablement has yet to be defined. For instance, in the POR example described previously, the login service could be reused across many business processes. However, different processes have different policies concerning the user class of service, that is, the type of authorization required. The virtualized IT infrastructure must be capable of ensuring different types of security, based on business policy, within the specific technology used to enable the application (for example, Web services).

Any solution development for on demand SOA must focus on an integrated approach among the following: the on demand business-process transformation that is driving application enablement, the corresponding IT policy and governance, and the system management of virtualized resources based on service level agreements (SLAs).

The starting point for on demand SOA is business process transformation. At the core of business process transformation is the policy and governance regarding how different parts of the process are integrated. This business process policy is used to derive IT policies, QoS, and SLAs. All of these terms are related to the business process, but each has a different meaning, as described below:

- *Policy*—A policy is a high-level statement of how things are managed or organized, including management goals, objectives, beliefs, and responsibilities. Policies are normally defined at an overall strategy level and can be related to a specific area, for example, security and management policies. In many instances, policies reflect the law or other mandated requirements to which the policies must adhere. This is especially true in the case of security and privacy policies.
- *SLA*—A service level agreement is an agreement between an IT service provider and the business that includes:
  - — Performance and capacity (such as end-user response times, business volumes, throughput rates, system sizing, and utilization levels)
  - — Availability (mean time between failure for all or parts of the system, disaster recovery mechanisms, mean time to recovery, etc.)
  - — Security (for example, response to systematic attempts to break into a system)
- *QoS*—Quality of Service addresses all features and characteristics of a product or service that bear on its ability to satisfy stated or implied objectives (from International Organization for Standardization [ISO] 8402).

Furthermore, the on demand SOA-solution design process must identify enabling technologies for various IT virtualization functions. Fundamentally, the on demand SOA solution must clearly articulate the approach for the following:

- Transforming business processes to be more dynamic and responsive.
- Rewriting or enabling applications with Web- or grid-services interfaces.
- Developing corresponding process-based policy and application awareness. Such changes will also make more effective the implementation of an on demand IT infrastructure with, for example, grid (schedulers, resource brokers, and federated file systems), autonomic (workload management), and utility (provisioning) resource management.

On demand SOA-related technology

Many key technologies and standards must be considered as part of an IT strategy if the goal is to become on demand through SOA.[6] Key enablers for on demand SOA include virtualization of the infrastructure and application automation of management. Enabling business processes, applications, and IT infrastructures cannot be a monolithic effort with overly rigid goals, such as turning every application in an enterprise into a Web or grid service, or completing all transformations by a specified date, or orchestrating all server, network, and storage resources with a single policy. However,

applying the key technologies for on demand SOA is a critical success criterion for achieving the benefits of on demand SOA. As discussed previously, these technologies include Web services, autonomic computing, utility computing, and grid computing.

CUSTOMER SCENARIO

In order to illustrate the concepts of on demand SOA solution development, we present a scenario derived from an actual case study with a large financial service company. We call our business process scenario "Life Change."

Business process description and customer goals

Consider a client service consisting of several processing steps that a customer must follow to modify employee benefits when a new child is added (by birth or adoption) to a customer's family. These steps includes health insurance additions, life insurance increases, and investment portfolio analysis. More specifically, the business process consists of the following fundamental subprocesses:

1. Change Form W-4 (U.S. federal income tax withholding) exemptions.
2. Add dependent medical coverage.
3. Add beneficiary to 401(k) retirement plan.
4. Open 529 account (college savings plan).
5. Run advice engine (using High Performance Computing [HPC] analytics) to suggest an investment strategy to achieve 529 goals.
6. Create payroll deduction to fund investment options for 529 plan and to adjust options for 401(k) plan.
7. Increase term life.
8. Run advice engine to suggest funding mechanism for term life, with minimum tax implications.
9. Suggest selling shares to fund term life increase and execute sell.
10. Run portfolio analysis including pension-plan estimate with various fund accounts to meet new long-term financial objectives.
11. Suggest rebalancing of portfolio and execute rebalance.

This overall process is shown in ***Figure 1***. Black arrows indicate the functions outlined above that are initiated from the customer's Human Resource's (HR) site. Red arrows indicate the functions that are initiated from the company's Web site. Blue arrows indicate functions that are now initiated by a phone call to an advisor and are asynchronous in nature. The light blue arrow indicates a third-party interaction.

The current implementation of this process consists of applications and resources distributed across not only different departments (for example, health benefits and investment benefits), but also resources outside the company (external investment funds). From an application point of view, these steps include stateful and stateless transactions, stateless analysis engines, and stateful data interactions. (Stateful implies the capability to maintain last-known or current status; stateless does not.) Furthermore, the process is currently supported by on-line subprocesses and off-line batch analysis and advice engines driven by asynchronous requests, including phone center requests. These flows are illustrated in Figure 1.

The current process implementation requires customers to follow a list of action items, connect to different Web sites or electronic forms, make phone calls, wait for data to be generated by various applications, and share data among the different applications. In our scenario, the client company has indicated that the current process results in a significant level of customer dissatisfaction as well as a number of customer requests that cannot be served by existing resources. This threatens competitiveness in the marketplace, and therefore the client is in danger of losing business in a highly competitive market.

The financial service company's primary goal is to find a low-cost solution to improve customer satisfaction and increase request throughput. The resulting higher efficiency from the improved process would in turn generate higher volumes of completed requests and drive additional business as more customers are served. More specifically, we can define the objectives of business process transformation for the Life Change scenario to be:

1. Use of a single identity for customers for both internal and external interactions
2. Collaboration with partners, with real-time interfaces replacing off-line batch interactions

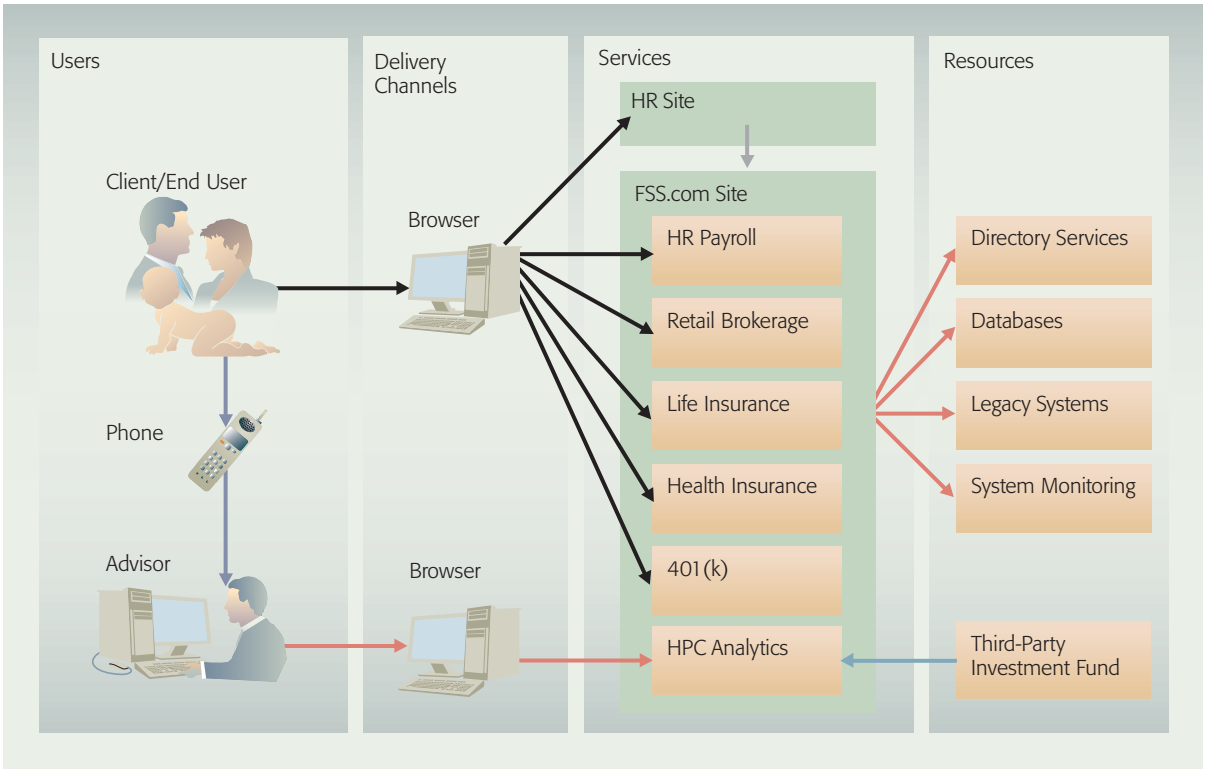

Figure 1
Architecture of the original process for a financial service company's Life Change scenario

3. Consolidation of many customer interfaces into a single, consistent customer interface
4. Dynamic allocation of system resources as types and volumes of customer requests change
5. Creation of a customer experience that allows a single combined view across all relationships, both internal and external

Process-driven meta-service composition and policy definition

From the list of steps given in the previous section and the functional flows shown in Figure 1, we can immediately see that within the coarse-grained Life Change business process there are subprocesses that will become services. Hence, we associate the overall Life Change business process with a corresponding composition of services or a *meta-service*. In this section, we demonstrate that each of these services has an associated policy, in terms of QoS or availability, necessary to meet the overall business process goals. Therefore, one way to evaluate our customer scenario is through the combined process decomposition into multiple services and the policy drivers for each service. Establishing this view of the customer scenario is vital because it is a key ingredient for our design work described in the sections that follow.

As part of policy-based management, control hierarchy and services must be described. The policy hierarchy is used to decompose business-process requirements into policy statements regarding the operational and functional characteristics of the services and underlying infrastructure. This is important because formerly functional and operational requirements were captured as part of a static architecture that was manually and iteratively tuned. With the adoption of on demand SOA, it becomes necessary to capture requirements as policy statements in human- and machine-readable formats. Thus, emerging technologies such as Web Service Level Agreements (WSLAs)[7] have been created.

Regardless of the implementation, it is important to understand the steps required to decompose a process into a series of meta-services, services, and operational characteristics. Process workflows and use cases are employed as a basis for defining the

composition of services. During the development of process workflows, functional and operational characteristics are captured including, for example, time and resource constraints, environmental issues, performance, capacity, security, and organizational issues. These characteristics, once documented, need to be implemented in an appropriate resource management schema. Process requirements are used in order to ensure that sufficient resources are provided to meet QoS requirements defined within SLAs. Global resource managers, in coordination with schedulers and workload managers, determine what hardware or operating environments are most suitable and available for a specific job or task. In addition, environmental requirements may dictate that a specific type of engine or configuration be provisioned or instantiated in order to reallocate resources based on a predictive analysis of workload trends or a manually triggered allocation or configuration change. These policies are defined as business, functional, and operational (nonfunctional) requirements within a typical on demand SOA.

A second set of policies defines guiding principles that are focused on management of the environment with respect to how the organization, processes, and tools are defined, developed, and built. This is required in order to sustain a well-operated environment. Separating the processing and data environments from the service-development environment introduces a series of issues that must be addressed. The separation of business, application, and IT policies also requires that various issues be correlated, rationalized, and arbitrated.

DESIGNING AN ON DEMAND SOA

There are multiple techniques that can be used to perform business transformation. Transformation implies that the re-engineering of process workflows does not end solely with a successful implementation of a system or education program. Rather, business transformation has many broader elements, including those handled by organization and change management competencies.

Previously, business strategy and analysis techniques focused on a process-centric model of a business. The process view did not always force the generalization of common tasks, nor did it aid in identification of "specialist service centers"—the building blocks of on demand computing. The emerging IBM-developed component business modeling (CBM) technique represents a business as a set of collaborating components that consume and provide business services. These business components are combined to form a viable operating model. Because SOA has implications for both the technology and the business levels, the transformation of CBM-identified business services into an on demand SOA brings more business focus to how we discover and publish SOA services.

On demand SOA requires the key elements of organizational transformation to be in place. Otherwise, the culture of the organization can continue to be a major obstacle in adopting new processes, technologies, or management techniques. In addition, an on demand transformation is much broader than any tactical initiative and needs full management leadership and commitment from the highest levels down. Dynamic process and technology sourcing decisions will not make employees comfortable without appropriate communication and education. As a result, business transformation projects are often put at risk, with employees potentially resisting any change and in some cases even acting as saboteurs to the adoption of new processes. In addition, technological constraints, whether purposefully implemented or due to current technology limitations, can impose barriers to meeting certain performance objectives. (Performance objectives are typically metrics incorporated in business processes in order to measure key attributes such as speed, cost, and accuracy of results.)

Business process integration is a business enabler which requires that an on demand SOA be in place in order to provide the necessary workflow infrastructure. Today, application development tools have evolved to a point where process life cycles can be enabled from start to finish. Moreover, process-modeling tools can automatically generate use cases and interaction models. These models can be dynamically instantiated with components provided as part of the application framework. Applications are then assembled from a collection of services that provide specific functionality. Ideally, this functionality becomes part of an extensive library that can be published as Web services, which in turn can be reused by others simply by discovering the service by means of the appropriate service directory. The benefit of this model is that it is based on the application of frameworks, which are being

improved on a daily basis to add increased breadth of packaged functionality. Admittedly, however, this somewhat utopian view of services still lies in the future on the business transformation roadmap. Indeed, the discussion clearly shows the large gap between what we have today and what is required in the future.

Highly dynamic application environments that are built upon readily instantiated services require an on demand SOA infrastructure that can

- support the dynamic allocation of resources needed to instantiate the service
- measure the service level characteristics of the service, and
- sustain service level agreements within tolerance by adjusting the infrastructure using available resources.

Furthermore, both integration of the underlying resources with these services and the enablement of corresponding IT management systems must be accomplished through the use of open standards. Applying open standards allows companies to maximize the benefit of various vendor component solutions and to ensure that these components will work together.

We now proceed to the steps required to design an on demand SOA solution. These steps will focus on the concepts of business process and policy as well as application and IT infrastructure enablement. More specifically, we need to understand the business process in the following contexts:

1. Current workflow definition
2. Compliance with industry standards
3. Componentization of workflows
4. Parameterization of workflows
5. Policy definitions within the workflow

After we understand the business process within these contexts, we can proceed with application and IT infrastructure enablement. Indeed, one of the goals for our solution framework is that the business process and policy drive the application and IT-infrastructure implementation.

Process and policy definition

The first step in building a solution for the financial services company's Life Change scenario is to clearly articulate the goals of the business-process transformation. As discussed, these goals can be summarized as follows:

1. Increase customer satisfaction with a more uniform and seamless approach to the Life Change business process.
2. Increase the throughput of available request processing by leveraging existing resources and improving their utilization.

Overall, the combination of these two goals will result in the financial service company remaining competitive in the marketplace.

We described the Life Change process as a series of steps executed to build a complete business process, and we also alluded to the fact that this process could be thought of as a meta-service composed of a set of service components in a workflow pattern. That workflow is shown in ***Figure 2***.

We now must define several overall process policies, which will in turn generate policies for each individual service. Let us assume that in order for the financial service company's implementation of the Life Change process to be competitive, the start-to-finish time for completion of a request must be less than 5 minutes. In addition, customers expect a 95 percent availability rate for the Life Change service (across all components), and are willing to pay more for increased availability or throughput. Finally, to reduce complexity the financial service company needs to present a single client-facing portal for all services within the Life Change process. We can therefore set the following process policies:

1. An end-to-end request must be completed in less than 5 minutes.
2. The supporting application and IT infrastructure should have a 95 percent availability target.
3. A Gold customer has precedence over a Silver customer, and a Silver customer has precedence over a Bronze customer. Customer classes are based upon a price/throughput algorithm.
4. All data feeds (data supplied by the customer/data sent to the customer) must go through a single, secure application portal.

After these high-level policies are set, we must derive policies for each of the services. From an

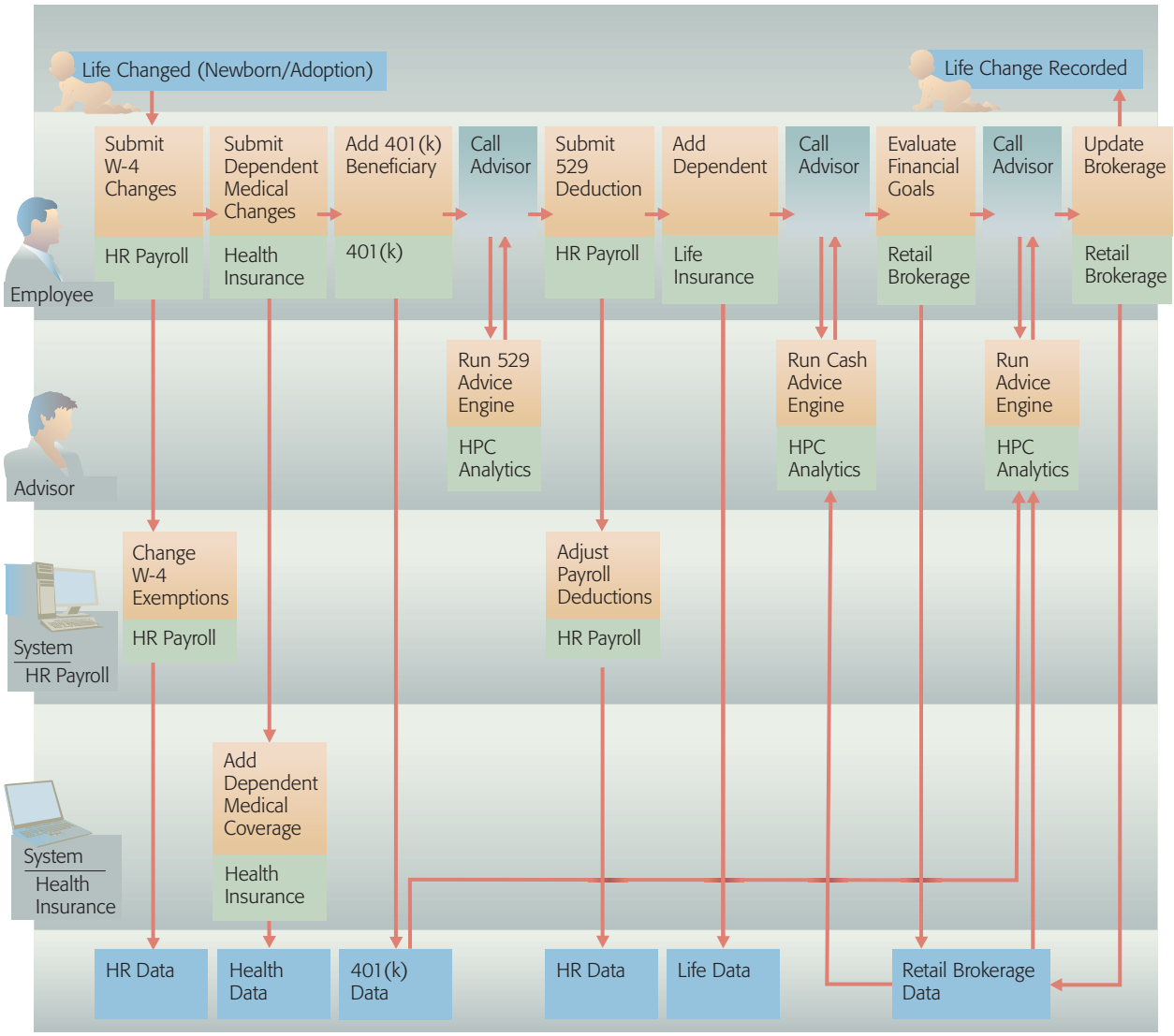

Figure 2
A flow diagram indicating subprocesses and databases for the Life Change process

availability aspect, we can make the simplifying assumption that if one service is not available, the entire process is unavailable; hence, we can state that each service must have an availability target of greater than 95 percent. The response time QoS will require a greater understanding of the complexity of the applications and the potential infrastructure used to run each of the services. For instance, we must determine a theoretical maximum response time for each of the services. We may decide that each transactional service must be completed in 20 seconds and the computation for the analytics must be completed in 2 minutes in order to reach our overall business process QoS. Establishing customer classes on a per-service basis to mediate requests when these objectives cannot be met is a further policy requirement.

There is a single point of entry and exit for data. Consequently, there is a single governing security policy for the process, and the individual services adhere to this policy simply by presenting data (for example, data used for analytics or account listings) through the secure channel provided by the initial login/authentication service. Thus, a process policy has actually driven a new service to be defined, namely a login/authentication service. We redefine the services workflow for the process accordingly

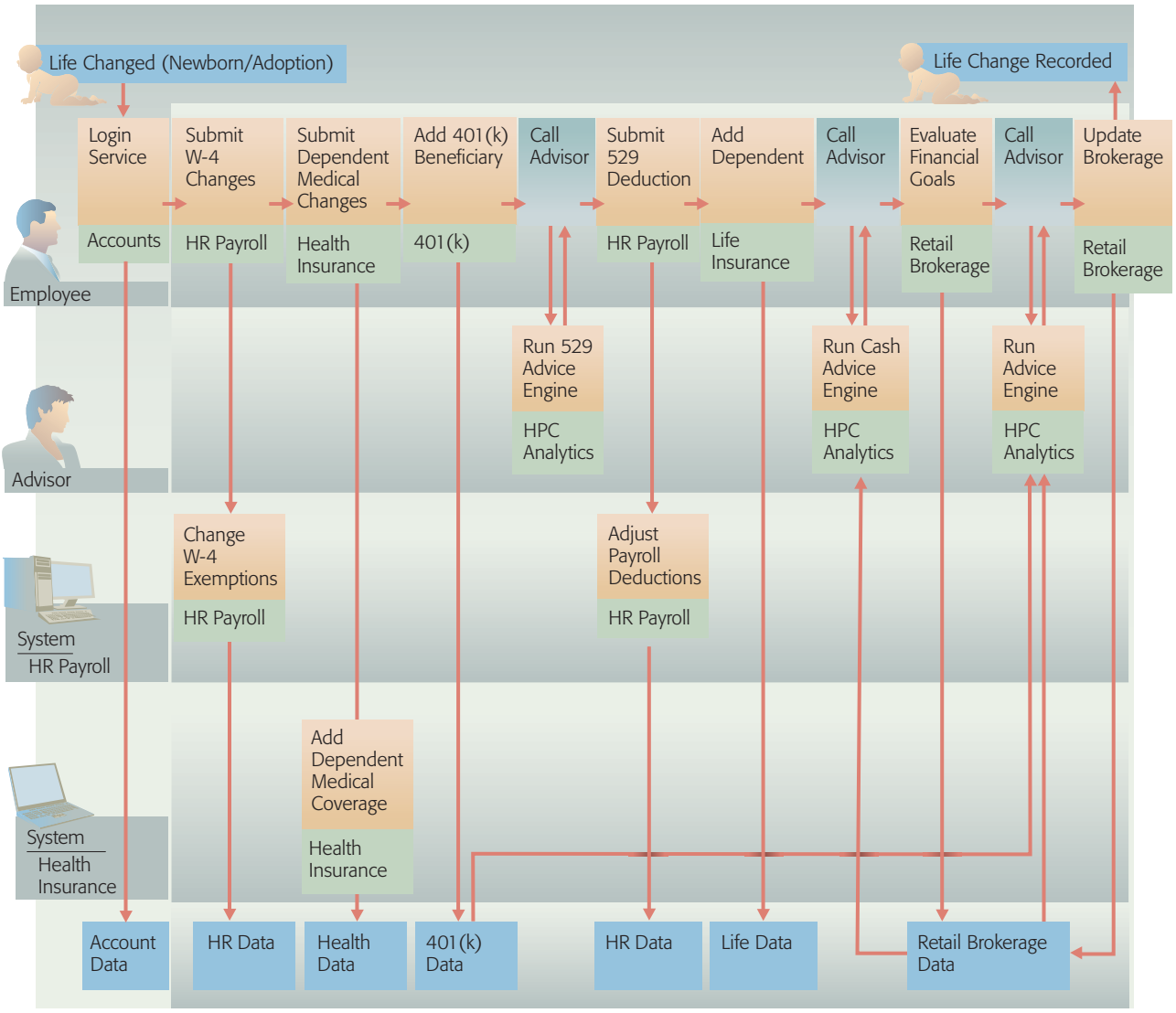

Figure 3
Refined flow diagram indicating subprocesses and databases for the Life Change process

and show the results in ***Figure 3***. It should now be clear why process and process policy must drive services and service policy definition.

Resource mapping

Now that the business process and corresponding services have been identified, we need to identify the resources, including software, hardware and change management, that will be involved in the re-engineering effort.

In the Life Change scenario, we need to specifically identify the resources associated with each of the services (both application and corresponding system infrastructure, for example, servers and databases), and map dependencies for each service. This is especially important in understanding how legacy transformation fits into application enablement. For instance, the applications corresponding to the HR payroll service may exist entirely on mainframe technology using underlying messaging middleware. We need to understand the effort required to move those applications and corresponding middleware to another hardware environment, such as a Linux** cluster. This approach calls for the maximum possible decoupling of applications from the underlying resources on which they currently operate. ***Figure 4*** shows a possible representation of

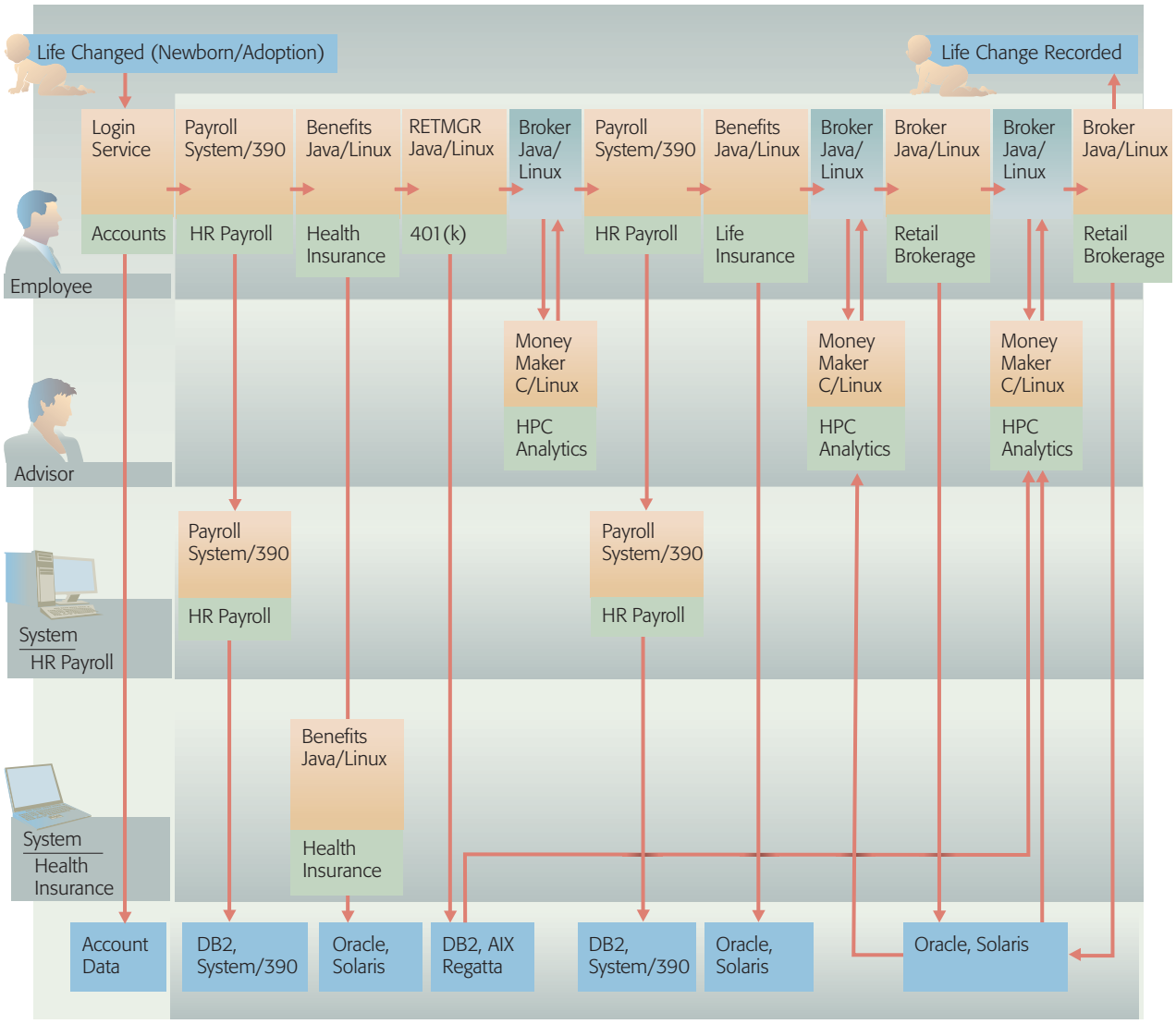

Figure 4
A resource mapping (applications and operating system) for the Life Change process corresponding to Figure 2

the current resource mapping, with an application definition and a hardware dependency shown for each service in the workflow. We have specifically added a login subprocess and a corresponding accounts database to comply with the single-point-of-entry policy.

Note that in the interest of brevity we have omitted a physical topological drawing of the actual IT resources and dependencies in Figure 4. However, analysis of physical topology is also a necessary step in the resource mapping to determine what actual physical resources will be affected, especially if the resources cross departments or lines of business.

Application enablement

At this point, the services and corresponding workflow have been identified. We have also mapped application functions to services that they will deliver. Now we must decide how to best encapsulate those applications in the SOA. At one extreme, we could completely re-engineer and rewrite an application both to make it independent of the hardware from the outset by using, for example,

J2EE** (Java 2 Platform, Enterprise Edition) and to make it easily published as a Web or grid service. However, this could be prohibitively expensive. At the other extreme, we might choose not to implement an application as a service, but instead continue to invoke the application in its current form with a service layer wrapper around the software. Clearly, this does not enable the application to run on every platform, but it may be sufficient to achieve the overall business process goals.

The application enablement environment for SOA

A fundamental concept of SOA is that a service is made available by publishing the interface specification. The application environment can then be used to create process-driven workflows that are easily reconfigurable through the dynamic integration provided by advanced business-process choreography engines coupled with standardized application interfaces. In addition, the application environment must provide a framework within which developers can easily determine when and how to abstract the data involved. Various repositories are linked together through these abstractions and corresponding data-sharing mechanisms. Applications can begin to use common interfaces in a common namespace (a set of names that is defined according to some naming convention). The discovery of the data location and its replication and caching is handled by the underlying infrastructure.

As shown in *Figure 5*, business services are composed of a variety of services that enable the overall business process. In a true on demand SOA, the applications themselves may be written on top of underlying application services and integration services.

After we have determined which applications provide function for services, we then have to decide at what layer to virtualize the corresponding function or collection of functions as a service. The level at which this virtualization is performed would ideally be determined by a compromise between compositional complexity (for example, building a process workflow out of tens of services as opposed to thousands) and maximizing code reuse. However, the reality of legacy application transformation dictates this does not always hold. For instance, in some legacy applications it may be impossible, or at least prohibitively expensive, to decompose an existing application into a collection of services. However, when re-engineering is feasible, we need to take the following design steps:

1. Determine a compositional or fundamental building block model (i.e., the features that are ubiquitous for given applications and the level at which abstractions should take place) with an emphasis on dynamic and reconfigurable building blocks.
2. Determine the distributed programming paradigm for each fundamental block (for example, data parallel versus control parallel or distributed memory versus shared memory).
3. Select a programming model for each building block.
4. Determine the statefulness of each building block.
5. Select the programming model for each compositional block and overall service, taking advantage of open standards for resource virtualization (e.g., Message Passing Interface, J2EE, SOAP, WSDL, and OGSA/Open Grid Services Infrastructure [OGSI]).

In our Life Change scenario we use the information provided by the resource mapping shown in Figure 4 to draw the following conclusions about application enablement:

1. The HR payroll service involves a single legacy HR application, which we are unable to decompose into lower-level services at this time; thus, a simple WSDL wrapper connecting to a Java**-RMI (remote method invocation) interface will be used.
2. In the interest of performance, the calculation for analytics should be virtualized at the parallel job level, not at the level of independent tasks within the job. In other words, the calculation should be considered as a single service executing on a virtualized server that consists of several computational nodes.
3. The login/authentication service is entirely new and stateless and therefore will be written using J2EE and WSDL interfaces, but will be based on some of the technology from the retail brokerage Hypertext Transfer Protocol (HTTP) client.
4. Because the life and health insurance applications have Java application programming interfaces (APIs), are stateful, and exchange

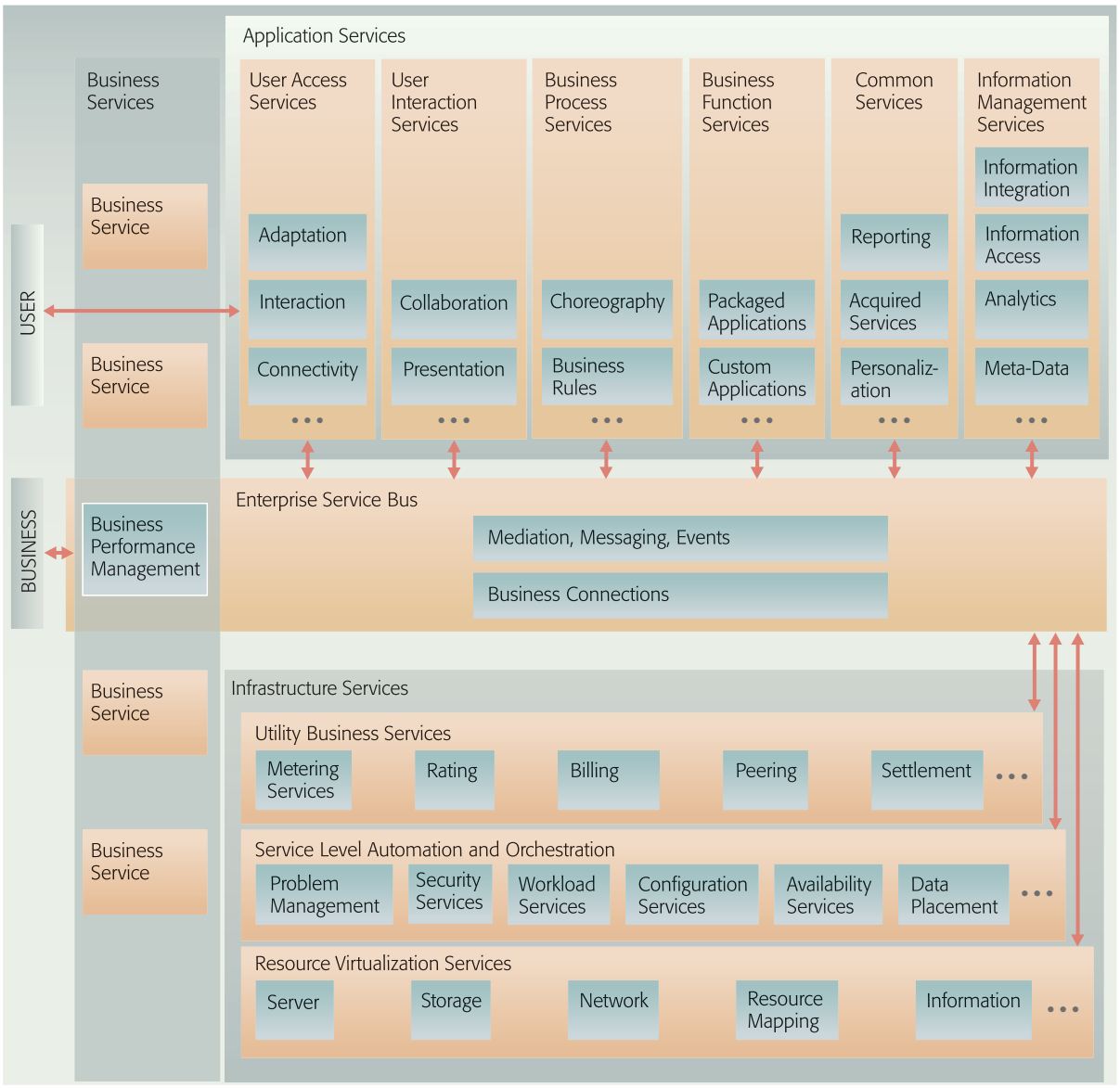

Figure 5
Architectural overview of IBM's on demand operating environment

transactional data, we will write an OGSI wrapper around these applications to enable them as services.

5. The existing retail brokerage and 401(k) J2EE code will be enhanced with OGSI interfaces, because those data are stateful and because transactional data are shared across both services.

Enterprise infrastructure enablement

A virtualized infrastructure is a fundamental step in implementing an on demand SOA. The on demand environment requires an infrastructure that can be dynamically adjusted based on new workloads and changes in the business process. However, before we enable the IT infrastructure for on demand computing, it is important to determine if the existing IT resources are sufficient to meet our business-process goals and policies. For instance, we must know whether there are enough nodes in the cluster that handles HPC analytics to deliver a response time of 2 minutes for an investment strategy calculation. After we have determined that

sufficient capacity exists in our infrastructure (and making the assumption that we will be operating in a virtualized environment with improved utilization), we can then begin to virtualize the physical resources.

A virtualized systems environment

The on demand systems environment provides an infrastructure of connectivity to resources on an as-needed basis. The infrastructure also provides more commonly reusable and dynamically instantiated containers that allow for ease of portability and common interprocess communication. The OGSA specifications enable these requirements to become reality by evolving Lightweight Directory Access Protocol (LDAP) standards for Web services discovery services. The core systems environment provides operating-system (OS) and hardware-level integration that enables the dynamic reconfiguration and partitioning of resources. Partitioning and reconfiguration are performed by a number of controllers, which can be built conforming to the OGSA standard. Virtual machines interlocked with OGSA-enabled provisioning systems completely abstract processors and storage from OS images. As this occurs, resource allocations are determined by service level objectives and subsequently correlated to metering data provided by complex and autonomic monitoring subsystems. These layers of virtualization and corresponding system management support are shown in Figure 5. OGSA provides the bridging between the systems and application environment, allowing for the virtualization of both systems resources and application meta-OS services.

Finally, security and support services need to be interlocked in order to maintain service integrity and provide access based on service subscription agreements. Lines of business must pay for what they use, and correspondingly, the infrastructure must be compensated to maintain quality.

IT management process and policy definition

Managing an on demand SOA begins with translating business process policies into enforceable IT policies to manage the various services in a virtualized environment. A strategy must be put in place defining the enablement of the various applications of the process in question as services. After services are defined, policies must be identified and defined because policies determine how an on demand business process should be orchestrated. Policies concerning privacy, security, authentication, resource-sharing priorities, accounting, and chargeback can be formalized through policy workshops and then applied using IT management models. We describe these concepts here.

Policy workshops. Policy definition is often conducted by using a well-documented technique known as a policy definition workshop. This workshop has its roots in what are also referred to as "guiding principles" workshops. The definition of policies in such a workshop is typically facilitated by the use of a predefined straw-man policy starter set. Policy workshops can be used to define the guiding principles within an organization that determine how the organization, processes, and tools are defined, developed, and built in order to sustain an on demand SOA. Key issues that are typically used as a starter set include the following:

- *Sharing*—What are the organizational rules of engagement when contention exists for resources?
- *Ownership*—Who maintains the financial burden for the asset, and how is the cost allocated, distributed, and recouped?
- *Charges*—What are the usage charges associated with the utilization of services? To what level of granularity is usage metered, and what is the associated overhead associated with fine-grained usage metering and billing?
- *Allocation*—How are these resources allocated, and how are priorities defined?
- *Commitments*—How are SLAs measured, and how is monitoring data aggregated and mathematically modeled in order to measure QoS or SLA attainment?
- *Schedules*—What are the rules associated with defining batch, maintenance, and other operational windows, particularly where functionality of a particular resource pool may shift in allocation or functional configuration?
- *Standards*—What level of standardization and interoperability is required for architectural elements?
- *Designs*—What is the general approach or priority for migrating legacy applications, analyzing the enterprise application portfolio, better leveraging HPC resources, and better leveraging transactional resources?
- *Services*—What core services must the environment provide in order to enable key management

and accounting functions? How will various resource managers communicate and publish their interfaces?

Management model. The management model defines the point of execution and control across a management infrastructure. The points of application execution and control must also be understood; however, in terms of management functions, a management model is instrumental in helping to institutionalize an on demand SOA. There are various forms of management model:

- *Centralized*—Central points of execution and central points of control are found in traditional legacy environments.
- *Enterprise hierarchical*—This model is typical of a large distributed environment where control must be centralized, but systems continue to execute functionality locally and communicate results upward to a single enterprise manager. This is similar to an intra-grid[8] across an enterprise.
- *Distributed hierarchical*—This model is suitable for a federated approach to management (distributed and collaborative control), allowing local domains a certain level of autonomy (distributed execution), while ensuring that there is cross-domain collaboration as required. This is similar to an extra-grid[8] or multiple intra-grids, wherein resources are managed as distinct domains, but may at some level participate in a collaboration.
- *Workgroup*: Islands of automation can be a viable and efficient solution in sufficiently simple situations.

Template-based process and service re-engineering. IT process re-engineering is often based on the IBM IT Process Model (ITPM), but may also use the IT Infrastructure Library (ITIL**).[9] These comprehensive models provide frameworks of activities grouped by relevant scope of processes. The various process outlines include such information as where processes begin and end, activities, inputs and outputs, measurements, dependencies on other processes and data, and what is included and excluded in the scope of each process. This methodology provides a template-based approach to designing processes based on prebuilt workflows. We assume that many of these workflows, such as problem management, can be nearly 80 percent complete and applicable in most customers' scenarios.

Adoption of new technologies will change service definitions in an on demand SOA. For example, the autonomic architecture defines a series of service flows that essentially couple various process activities in order to provide some end result to a customer of IT. Thus, change deployment can be coupled with other process activities to create a service, for example, when it becomes part of a provisioning service. Provisioning, when defined as a service, can and will include elements of configuration management and change management, among other process activities. The adoption of service definitions will become more critical as on demand SOAs drive the need for more orchestrated cost and resource management. Users will truly become subscribers to an on demand SOA, requiring well-defined and measured services that deliver a suite of capabilities. The intent is to decouple users and applications from an abstracted SOA framework and provide resources dynamically to address the QoS and SLA requirements of the business.

Virtualized infrastructure design

The guiding principles for a virtualized infrastructure are founded on policy-based resource sharing. Of course, the level at which resources are shared, from a logical partition (LPAR) to a cluster, depends on both the applications that use the shared resources and the organizational governance from which the policies are derived. For instance, in our customer scenario both mission-critical e-business applications and applications that use HPC analytics can run on Linux clusters. However, the level at which one department can charge for use of those resources (for example, on a cluster basis as opposed to a per-node basis) will help determine whether the resource should be virtualized at the cluster or node level. In fact, the underlying system and workload management of that cluster may only be capable of a specific level of resource-sharing granularity. Before a virtualized infrastructure design is begun, these types of constraints must be documented. Even more fundamentally, before undertaking the effort to consolidate resources for a virtualized infrastructure, some capacity planning and analysis is required. This planning will determine if enough resources exist to meet projected demand based on policies. This can be accom-

plished through continual online modeling and analysis of different systems.

Scheduling meta-services. As stated earlier, in an SOA design a business process can be described as a composition of several services, referred to as a meta-service. A request from an end user to execute a meta-service must have some initial policy-based scheduling, most likely based on user class of service and security-based policies. For instance, some customers in our scenario could be paying higher prices to ensure better Life Change process throughput; these Gold class customers would be preferentially scheduled over Bronze class customers. The meta-services scheduler needs to decompose the meta-service description of several participating services and identify a workflow or a calendar plan of how these services should be executed. This would include synchronous versus asynchronous service-scheduling constraints. After the plan is in place, the scheduler routes the service to the appropriate workload domain for scheduling and instantiation on the local resources. A workload domain consists of a request scheduler or router and a collection of resources. In certain cases multiple workload types can be scheduled within the same workload domain. For instance, in our Life Change scenario the WebSphere* services[10] would be routed to a WebSphere Web services scheduler, and the HPC application would be routed to the appropriate HPC scheduler. Data and states that must be shared by the various services within the workflow can be maintained within the meta-services scheduler. (Some SOA standards such as OGSI allow for state notification and data sharing, so this function would not necessarily always reside within a meta-services scheduler.) We also note that the meta-services scheduler could act as a single point of entry for all resource requests (services-based or otherwise) on the system, and this scheduler would then be responsible for routing a request to the appropriate workload domain for local resource scheduling. It is important to recognize that when working with distributed data, the SOA paradigm does not make data storage visible directly, but instead uses the concept of a meta-service to mask the physical data representation.

Policy-based resource sharing. Policies for resource sharing can be as simple as those based on time of day, or they can be as complex as those guaranteeing a certain level of performance or ability for a given service. In order to meet increases in demand for a given service, resources from other services must typically be reallocated. The corresponding optimization problem for autonomic resource management across multiple workload domains is quite daunting. As a first step, however, we can rely on the human decision-making process to decide when to move resources based on QoS guarantees and QoS violation events. Each of the actual deallocation and allocation processes can be automated by supplying tools and workflows for reconfiguring the available resources. We may refer to this as an enterprise, global, or multiworkload domain view of resource sharing. However, in order to develop a comprehensive workload management system, we need to further address additional levels of granularity within domains and resources.

In addition to the enterprise scenario just described, we must also consider policy-based resource sharing within a single workload domain, for example, a WebSphere cluster, a LoadLeveler* cluster, or a Cisco cluster. We are concerned with how requests from different users for a certain service should be handled based on incoming order, customer type, and current QoS statistics. If services of different types are managed within a single workload domain, that is, within a single WebSphere cluster, we must also consider how resources should be allocated for services of different types. For workloads of different types, the actual scheduling or load-balancing algorithms could differ greatly. For example, in HPC parallel jobs the scheduling requirements include resource matching, advanced reservation, and backfill scheduling for batch jobs. Interactive e-business or HTTP request load balancers may not be concerned with advanced reservation or backfill. In a resource-sharing system, both schedulers will need some user-based policies such as QoS and security. These policies must be built into each workload management system that they impact within the domain. We may refer to this type of resource sharing as policy-based scheduling or load balancing within a workload domain.

Finally, resources within a single machine can be allocated and managed according to a given set of policies. Operating-system-level controls, LPAR configuration, job swapping, and paging can be used to manage workload priorities according to a set of policies within a server. Memory and CPU resources

can be moved across LPARs using hypervisor controls, and jobs within a single LPAR can be halted while jobs of higher priority are swapped in to obtain the required system resources. Of course, the priorities of jobs and applications are set by the service policies. We note here that in all these cases the mapping of service policies to the actual tuning knobs on a system, whether that system be a single server or a collection of clusters on a network, may not always be straightforward and may require some additional tooling to implement.

We now need to relate this infrastructure to our customer scenario. Recalling the flows shown in Figure 4, we see that the 401(k)-plan, health- and life-insurance, retail-brokerage, and HPC-analytics services can all run on Linux clusters. We can realistically divide these services between two workload domains: an interactive, HPC parallel service workload and a Web services e-business multitiered transactional workload. Our meta-services scheduler for the cluster is quite simply a scheduler that knows how to route the different requests to different domain schedulers. The meta-services scheduler is also a multidomain resource manager that builds an initial set of nodes and dynamically manages the resources allocated between the two workloads to satisfy the policy constraints, most probably based on QoS or performance, for the different services. The dynamic, autonomic management of these resources between two workloads is a very difficult task and a topic of intense research today. We note that the meta-services scheduler would also be responsible for routing requests to the mainframe for the Form W-4-based service.

SLA-based monitoring and metering. In the preceding section, we reviewed requirements for policy-based resource sharing. This type of resource allocation is impossible without the appropriate monitoring and event infrastructure in place. SLA-based monitoring must be based upon service level guarantees and QoS violations. Furthermore, SLA-based monitoring is a requirement for accurate accounting, billing, and chargeback, especially when we view pricing as a type of policy for the system. In addition to monitoring the metrics specified in a given SLA, various components of virtualized resources must also be monitored, so that predictive models or analysis can provide information that can be used to better manage future allocations of resources. This type of analysis and allocation enables the system to anticipate SLA-based violations before they occur and take prescribed preventative measures.

■ Ideally, resources should be dynamically allocated not just according to demand but also in advance of the peaks and valleys of service requests ■

Another important aspect of this type of monitoring relates to transactional monitoring and dependency analysis. Consider that several of our services are transactional in nature and that the underpinnings of these services involve a traditional two-tiered architecture with an application server and a database server. This is the case with the benefits and retail-brokerage services in our scenario. In order to determine how to best allocate resources for these services after the QoS-based policies have been violated, we will need to know if more application servers or more database servers are required to meet our QoS goals. Bottleneck identification is a key for appropriate resource management in an on demand SOA for transactional applications. Often, the topology of a service request is not known, and dependency analysis is required to determine how many tiers are involved in processing a request. Therefore, dynamic instrumentation of services to identify bottlenecks must be available.

Accounting and billing. In any virtualized on demand SOA infrastructure, accurate accounting and billing is required. For the scenario in this paper, Linux cluster resources are shared among the analytics department, the retail brokerage group, and the benefits businesses. Beyond the meta-service request for which we must provide accurate metering information for proper billing of customers, these shared resources may also be used independent of the meta-service request. For example, the analytics department may want to use the combined Linux resources, when available, to run larger and more complex value-at-risk simulations for portfolio analysis. They would be using the

resources of another department and organization to do this. We need to account for the fact that these resources are not being used to support the Life Change business process (or any other business process running in the on demand SOA environment) and identify which department is using them and for how long. Billing or chargeback for shared resources is feasible using this type of information.

Federated data. Thus far, we have concerned ourselves with the sharing of computing or server resources. However, most on demand SOA scenarios, including our scenario, have data-sharing requirements as well. To understand this, we need only think about how portfolio management is done. In our calculations, we must consider investments in multiple funds, including 401(k), 529, and actual stock portfolios. Most often, these data sets exist on different departmental databases. When an advice engine is required, a person must then access disparate servers, collect the data by running a series of complex queries, and use the aggregate data to generate some investment advice. If the data were easily accessible programmatically by each of the services, the need for human intervention in this process would be removed. The architecture by which data is easily accessible in this fashion is called data federation, and, in this case, databases are federated. File-system federation most often occurs for numerically intensive computing jobs where data I/O has extensive performance requirements. In this case data, which often includes intermediate results of a calculation, are stored in files that must be accessible to all nodes in a distributed system, including access across clusters. For our scenario, the data in these files may also include state information or checkpoints used to save the state for each of the portfolio calculations within the job.

Network provisioning. Network allocation is also a key consideration for distributed-system and shared-resource management. Because some customers might have higher priority than others, we need to implement policy-based bandwidth allocation on the network, both into the main site (portal) and among and within the different workload domains as requests move from one service to another in the workflow. When nodes are allocated and deallocated, network provisioning is also very important to ensure secure access and appropriate network topologies for many multitiered transactional services. Network provisioning is important as internal utilities begin providing hosting services for external utilities. Virtual local area networks (VLAN) are one example of network resources that are configured through switches to provide data and server security.

Capacity planning. Capacity planning is critical to any distributed system design. In many respects, capacity planning in the on demand SOA architecture is similar to classic IT performance analysis work. There are, however, new business-process and IT-policy implications for capacity planning for an on demand SOA.

Business processes, applications, and the underlying IT infrastructure of our solution are on demand by design. This means that, ideally, resources should be dynamically allocated not just according to demand, but also in advance of the peaks and valleys of service requests. For instance, we would like to allocate servers at different times of the day to handle the Life Change scenario transactional services when the request rate is highest, but perhaps reallocate some Linux servers for different types of HPC analytics that could run overnight to help provide more information for accurate portfolio balancing the following day. How far in advance and with what accuracy these peaks and valleys need to be predicted are determined by IT QoS policy. Fundamentally, these changes in the prediction horizon interval correspond to a real-time analysis requirement for on demand computing. This means that we may need to run closed-loop, feedback-control predictions every 5 minutes; these predictions may not need to be extremely accurate, but they must be accurate enough to prevent overprovisioning or thrashing (unnecessary provisioning and deprovisioning of resources at excessive rates). We may still choose to run highly accurate, off-line, traditional capacity-planning algorithms at the end of every day, once a week, or once a month to aid in on-line prediction accuracy or simply to assess the efficiency of our current system architecture. We want our capacity-planning element in the on demand SOA to be capable of both on-line and off-line analysis and prediction.

Capacity planning in the SOA environment also requires that we be able to provide performance analysis and prediction for services, not just for single applications or transactions. This philosoph-

ical shift requires that a services dependency or topology diagram be made available, either through dynamic generation via traces or through static user description. These are, of course, two extremes of a spectrum of possibilities, and we envision that there will be ways to generate dependency diagrams that are a combination of user-defined topology and dynamic discovery of resources within a service. The services dependency diagram will correspond to an IT topology diagram, mapping services to resources. For instance, the health insurance subscription service corresponds to a two-tiered architecture with both application and database servers. Capacity planning for that service involves planning for both types of server capacity. After the IT topology for a service is known and the desired values for analysis are entered, the requisite measurements, including type and frequency, must be determined.

■ Because the on demand SOA philosophy is based upon open standards, implementations and future projections will rely on open standards roadmaps ■

Finally, when on demand SOA solutions include a grid implementation, the problem often referred to as "moving resources" exists. By this we mean that resources (computing, network, and storage) in a grid can join and leave a grid in a nondeterministic fashion. Therefore, we do not always know when capacity is growing or shrinking in our distributed system. We note that although this may seem like an intractable problem, most intra-grids are built with fairly strict IT policy about when and how nodes can join or leave a grid within the on demand SOA. This is true because business process, policy, and governance drive the on demand SOA. Thus, the risk of not knowing when resources are available for a grid is mitigated by the underlying policies.

A summary of the on demand SOA solution architecture is shown in *Figure 6*. Black indicates service requests generated by the user. Orange indicates data flow (synchronous or asynchronous) from a service to the user, redirected through the FSS.com site. Blue indicates monitored, SLA-based data collected at the different schedulers or load balancers and sent back to the resource manager; the resource manager also provides this and perhaps other data to the capacity-planning tool. The capacity-planning component provides SLA-based events or threshold analysis to the resource manager, so that the manager can either reconfigure existing allocations to keep up with demand before peaks or valleys occur or configure the metascheduler to offload a request, if appropriate, to an available third-party hosting service shown in purple.

IMPLEMENTING AN ON DEMAND SOA SOLUTION

The final steps in our hypothetical on demand SOA customer engagement would be first to implement our solution as thoroughly as possible with current products and then to indicate where these products are lacking and provide a reasonable technical roadmap for achieving a more complete solution. As indicated previously, to see real benefit from this approach we cannot afford either a solution requiring complete elimination of existing resources or a solution in which all of the applications and all of the IT infrastructure must be transformed at once. Furthermore, because we are developing solutions for an evolving technology, we are necessarily limited to what is already existing and mature within distributed-computing and Web-services product bases. Because the on demand SOA philosophy is based on open standards, implementations and future projections will rely on open standards roadmaps. Finally, we note that in this section we are proposing a set of technologies that are derived from autonomic, grid, and utility-computing technologies. There may also be other viable technologies and solutions. However, our purpose here is primarily to demonstrate by specific examples how one could implement an on demand SOA for our given customer scenario.

As has been the theme in this paper, we describe the technologies in our implementation in reference to business process, application, and infrastructure enablement. We also identify specific autonomic, utility, and grid-computing technologies where appropriate.

Business process enablement

Developing an overall on demand corporate business strategy will remain the work of experienced business consultants who have an in-depth knowledge of the emerging enabling technologies and

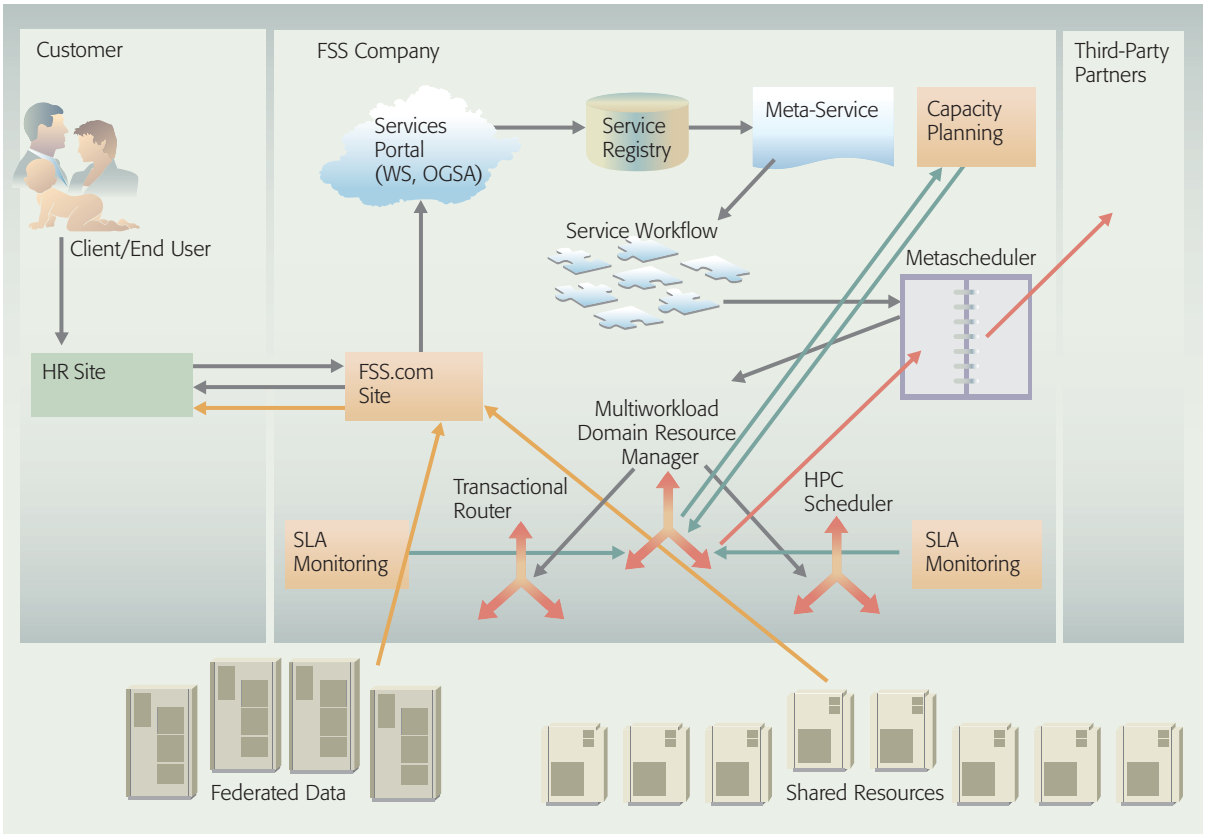

Figure 6
Architecture of an on demand SOA solution for a financial service company's Life Change scenario

open standards. Once an overall on demand business strategy has been developed, specific business processes that have been identified for on demand SOA enablement should be addressed within the context of a certain core group of technologies.

One specific technology for business process enablement is the language used to describe a business process or meta-service. Today, Business Process Execution Language for Web Services (BPEL4WS[11] or BPEL for short) is an obvious choice, because it is standards-based and because such BPEL4WS-supporting technologies as composition tools and service decomposition engines already exist. In our scenario, BPEL4WS could quite easily be used to describe our meta-service as a collection of services, and our metascheduler would then just need a BPEL4WS engine to decompose and parse the meta-service definition.

Business processes are enabled at the IT level by using policy-based resource management. In this regard, it is important that the different IT policy descriptions, including enterprise workload management (eWLM[12]) and WSLA, be understood at a sufficient level that the business process policy does not require levels of IT policy and capability that do not exist or cannot be achieved.

Application enablement

As stated previously, the key decision for any services designed in an on demand SOA is to decide at what layer different applications should be virtualized as services. This type of analysis should become part of existing legacy-transformation and application-portfolio-analysis assets and offerings. However, after this analysis has been carried out, the next real decision involves whether to use Web services or grid technologies and standards, in other words WSDL or OGSI. The issue of whether to use WSDL or OGSI at this point is a very subtle one, and one that we expect to become more difficult as the two standards begin to converge. For our scenario, we may choose to do something like this:

1. For the services that change portfolio or investment profiles (retail brokerage, 529 account creation, 401(k) plan management), we will write the services by using OGSI and taking advantage of its notification mechanisms.

2. For the HR application, for benefits (life and health insurance) and for numerically intensive computing, we will use WSDL.

WebSphere provides a framework for developing and building WSDL-based services called WebSphere Application Developer (WSAD). There is no such equivalent in Globus Toolkit 3** for building OGSI-based services.

After a service has been built by using OGSI or WSDL, a container is required on each node on which the service will be instantiated. For Web services, this is typically WebSphere Application Server, BEA WebLogic**,[13] or Tomcat.[14] OGSI containers will be available on all IBM eServer* platforms. In the meantime, particularly for non-IBM platforms, one can use any SOAP container, for example, Tomcat. Again, the choice of which container to use is a fairly subtle decision based upon available middleware budget, support, and other considerations. For our scenario, we use WebSphere Application Server to develop WSDL applications, minimally implement Tomcat on every node in the system for containers, and have WebSphere Application Server installed on the mainframe.

IT infrastructure enablement

We now review the functional blocks from Figure 6 and describe the corresponding list of candidate technologies from autonomic, utility, and grid computing that could be implemented to meet the design requirements of our scenario.

Scheduling

First, we assume that BPEL will be the language used to describe the meta-service. Therefore, we will need a technology based on a BPEL4WS engine. Second, we will need a scheduler that has fairly rich APIs to accommodate the domain-specific workload balancers and schedulers for a specific type of service. Third, the meta-service scheduler should work across administrative domains as well. The Globus Resource Allocation Manager (GRAM)[15] provides a rich set of standardized APIs for various schedulers, so that workload domain-specific resources can be discovered and used by the appropriate applications. However, GRAM does not support a BPEL4WS engine. The DataSynapse GridServer** broker manager[16] provides service-based scheduling across many resource domains, but it also lacks a BPEL4WS engine and has no policy-based scheduling. Web Services Management Middleware (WSMM),[17] a technology developed in IBM Research and now released in product form, is a Web services scheduler, but it is unclear at this time how such technology would support multiple WebSphere Application Server clusters. Not enough is known publicly about the Platform Symphony** product[18] or the recently announced Community Scheduler Framework (CSF),[19] which will be an open-source metascheduler. Thus, at this point there could be three reasonable solutions for the metascheduler in our scenario:

1. Customers write their own scheduler with an underlying BPEL4WS engine and GRAM infrastructure.
2. Customers write their own BPEL4WS and policy-based scheduler and use DataSynapse to schedule services to the different workload domains.
3. Customers use a prototype based upon WSMM technology, including multidomain support.

For each of the workload domains we also need a scheduler. For the transactional workloads (everything but the HPC analytics service), we could use WSMM technology as is. WSMM is a good choice because it includes some resource affinity as well as SLA-based scheduling. For the HPC scheduler, there are many choices including Platform LSF,[20] Condor,[21] and OpenPBS,[22] among others. Platform LSF is a sensible choice because of its existing Linux and multiplatform support; however, its licensing fees may be a concern. OpenPBS has a Linux and multicluster solution, but our customer will probably not want to implement an open-source solution for such a key middleware product as scheduling in a mission-critical deployment. Another option would be IBM's LoadLeveler (LL),[23] which at this writing is under limited availability on Linux; additional work would have to be done for multicluster support. DataSynapse might be another choice if the analytics application fits well within the

DataSynapse programming paradigm. Platform Symphony actually has all of the function needed to schedule parallel services such as the HPC analytics.

There is also another possible solution to the HPC scheduling and parallel service problem. IBM has an existing parallel application programming environment called Parallel Operating Environment (POE).[24] POE represents a distributed parallel job with one context in a UNIX** or Linux operating environment. Today, POE is a runtime library that contacts a resource manager, in this case LL, to acquire a certain number of nodes with specified requirements to execute a parallel job on a system. POE provides wrappers around the main execution of the code on each node to start up and terminate jobs. In order for an HPC application to be treated as a service, POE would now need a Java wrapper to publish some type of WSDL for the service. Furthermore, if POE did not terminate the job after the main method of the code was finished, there would also be a mechanism to stream new data into the application code without restarting the job.

Given this background, we could now start up the parallel service using this modified POE and the LL resource manager. After that has been done, we need only a single service scheduler for the Linux clusters because the POE parallel service appears to the service scheduler as just another service on a virtualized server (a collection of Linux nodes used to execute a parallel job). The task-level parallelism within the job can be handled within the application using standard master-slave parallel-programming-paradigm implementations.

Policy-based resource sharing

In our scenario, we are most concerned with how the resources on the Linux clusters will be shared for the health- and life-insurance, retail-brokerage, and HPC-analytics services. If we know that one service is more important than the others with respect to ensuring that the Life Change process executes with reasonable QoS, then we can give that service preferential access to resources. We would need to implement this type of preferential policy first within a server and then across the distributed system.

Within a node, we can use eWLM to manage resources preferentially. Within a Linux cluster we can use WSMM policy-based scheduling to make sure that resources are scheduled optimally to meet different Web services SLA metrics, such as throughput. Finally, if the services can be split into different types of workload domains, such as the HPC domain and the transactional domain, or even into dedicated service nodes within the transactional domain, we can use IBM Tivoli Intelligent ThinkDynamic* Orchestrator[25] connected to an SLA-based monitoring and event system to dynamically reallocate cluster resources. Furthermore, some feedback between these different components will be required for more coordinated resource sharing. At the time of writing, however, the policy languages used by eWLM, WSMM, and ThinkDynamic Orchestrator are all different.

■ The overriding value of an on demand SOA is the ability to build once and use often ■

SLA-based monitoring and metering

Before we decide what SLA-based monitoring we will employ on our system, we have to decide what products are using SLA-based monitors for resource sharing and for accounting and billing. As we stated previously, the resource-sharing infrastructure components actually use a myriad of policy languages, and all provide relevant monitoring information for accounting and billing. One approach here would be to provide a central SLA-based monitoring manager that could actually connect to all of the data being generated by the resource manager agents. This manager would then also be responsible for sending events to a variety of subscribers, extending from the resource managers, schedulers, and provisioners to the accounting and billing functions. Currently, there is a system under development at IBM Research, designated the WSLA manager,[26] which provides just such an abstraction. This system is configurable to connect to any data source for monitored data, provided that interfaces to the data source are built and that they send data to any listener subscribing to SLA-based events. Work with the customer would involve writing the correct interfaces for data and events for customer-specific systems.

Accounting and billing

Accounting and billing are highly dependent on functions such as monitoring, which are required to

manage a utility. Fundamentally, data for accounting is based on usage metrics captured from monitoring subsystems. Monitoring systems tied to SLA management components determine if a target level of service is being breached; however, the same data can be tied into a usage accounting system to correlate applications with the systems on which they are running and with the duration of the run.

Accounting functionality will continue to evolve over time as probes become less intrusive and as measurement overhead is reduced. This will enable greater granularity in the data points that can be captured relative to application cycle counts over a period of time. Currently when an application begins to execute on a given server, a trigger indicates that work has started on that server and begins the billing period for a particular user of a particular resource. This information can also be obtained through historical analysis of such data as system usage logs. Software exists today that can be configured to address such accounting requirements, including SAS** IT Resource Management,[27] Evident Enterprise,[28] and Tivoli Decision Support for OS/390*.[29]

Billing and chargeback are based on the information found in accounting systems, including both usage data and the cost models and mechanisms by which the cost is allocated. Billing systems have existed for many years and will not require major overhauls. However, these systems will need to undergo a transformation in terms of how they are used in chargeback schemas within IT providers to ensure that costs are properly allocated to the correct subscribers based on the usage data from accounting systems.

Measurement data is usually available from several existing sources. For instance in the OS/390 environment, tools already in place will likely be System Management Facility (SMF), online transaction monitors, and database logging. For the UNIX and LAN/WAN (local area network/wide area network) environments, tools that may already be in place that provide measurement data are Computer Associates (CA) Unicenter** NeuMICs**,[30] Hewlett-Packard OpenView**,[31] or BMC PATROL**.[32] Implementation considerations include requirements for these tools to interface with or feed data to the IT accounting software.

Commonly used IT accounting or chargeback-specific products currently include the following:

- SAS IT Resource Management and IT Charge Management[33]
- CA MICS (NeuMICS) Accounting and Chargeback
- CIMS[34]
- Tivoli Decision Support for OS/390 Accounting and Accounting WorkStation
- Evident Enterprise
- Veritas MicroMeasure**[35]
- ADC Singl.eView**[36]

Federated data

We are concerned with two types of data federation in our scenario. First, the HPC analytics application could require shared data via a file system. In this case we are concerned with cluster file systems. In the Linux cluster market, IBM's General Parallel File System (GPFS)[37] is "state of the art" and the most affordable option available. GPFS is also being extended to work over wide area networks so that multicluster support is available. We note that several customers who are considering using HPC applications within SOA also wish to save the state between job execution cycles, for example, saving intermediate computational results in a type of distributed cache. Currently work is underway to investigate GPFS as the basis of a distributed cache.[38] However, at the time of writing no solution exists in the marketplace for the distributed-cache-in-a-cluster problem that performs well enough and is general enough to be part of an HPC solution.

The second type of data federation deals with database-type storage, allowing data to be shared across tables and even physical volumes. In this arena, much of the technology is still under development, but IBM does have a significant existing product in DB2* Information Integrator.[39]

Network provisioning

Policy-based resource sharing ties together a series of management components in order to reallocate cluster or server resources and application components or workload. Network provisioning couples policy-based resource sharing with an underlying dynamic connectivity environment to perform both topology reconfiguration and policy reconfiguration.

Intelligent networks today allow for the dynamic construction and deconstruction of private VLANs to

create logical segments. In addition, virtual private networks (VPNs) encompassing wide areas can now be used to secure bridges between private segments and allow collaboration within a public network. These services can be set up as required by network administrators to control traffic flow through private and public networks, and in our scenario could be used to build virtual Linux clusters for different services. This is more time- and cost-effective than creating clusters on truly distinct LANs. However, these services alone do not provide the dynamic provisioning required to ensure that the QoS requirements for application connectivity are met.

Congestion management is a more recent form of network provisioning. Many firms have begun performing QoS management by using tools from companies like Converged Access[40] or Packeteer[41] to manage congestion. These tools typically address the scenario wherein a firm knows it has more traffic than it can send over the current network and must therefore control which users or applications are given priority and which are denied service based on management policy. Networks tend to discard excess traffic (traffic above a certain threshold). QoS management systems enable the network to make decisions based on QoS reporting of how effectively policies are being attained. Typically these systems provide a probe in-line between the WAN and border routers to manage congestion. Products such as those from Converged Access or Packeteer provide devices or probes that measure QoS to rearrange traffic. These products manage defined traffic flows and also tell the administrator about unknown traffic flows in addition to those that are configured. This process relies on well-specified input in terms of policy and traffic flow analysis.

Over time it is expected that QoS and advanced reservation protocols will become more commonplace and will be embedded within vendor hardware and operating systems in order to ensure that applications can be provided connectivity at a certain guaranteed level of service. In the meantime, point solutions exist that can be coupled together to begin orchestrating allocation of bandwidth to applications.

Capacity planning

As stated previously, capacity planning for on demand SOA should be based on services. This means that we will need dependency analysis and corresponding IT-based topology maps for each service. In our scenario, we will need to understand the request path or server topology for each of the services. For instance, we will need to know which servers are required for an "add dependent medical coverage" service and how these servers are connected (e.g., application server and database server). The eWLM policy description provides the required dependency map. However, in order for requests within a service to be traced with eWLM, the application or associated middleware should be instrumented with application response measurement (ARM) calls. For some applications, this is not possible. New technology for dynamic dependency analysis that does not require such instrumentation is under development.[42] Even this newer technology has some limitations, however, because it requires that applications be written in Java. For non-Java applications, building dependency and topology maps can be much more difficult, and experience in analyzing existing application logging becomes crucial.

After the topology and dependencies are known for one of the services, we would specify a certain QoS, in terms of either throughput or response time, and generate an appropriate system configuration to support that QoS. Combining these capacity requirements is an optimization problem. In particular, analyzing possible workload scenarios is crucial to addressing both the advanced reservation problem for any policy-based resource manager and the dynamic provisioning of resources. In our scenario, we would use such tooling to configure the system one way for a time of day when we expect a heavy customer load, perhaps allocating more Linux servers to the advice engine application. Then we could reconfigure the system to give more capacity to the other Linux applications during off-peak hours. Sophisticated capacity planning and optimization models are clearly required for these scenarios. An example of such technology for advanced tooling that combines request topology, use prediction, performance analysis, and resource optimization can be found in Reference 43.

CONCLUSIONS

The overriding value of an on demand SOA is the ability to build once and use often. Moreover, it is an opportunity to bring the concept of encapsulation

from the object world, where the reuse impact is small, to the enterprise world, where the impact will be far greater. In on demand SOA, integration is also explicitly defined and better understood at the application and enterprise level than can be the case in current environments. Overall, we can expect that on demand SOA will lower development, operations, and maintenance costs. Finally, linking requestors and providers by contract agreements will provide a looser coupling in which each can vary the specific details of his or her own work independently.

In this paper we have used a specific customer scenario to demonstrate how an on demand SOA allows us both to define a business process explicitly and to separate it from the definition of its individual component steps. One important result is that multiple processes can now share the same implementation of common individual steps. In particular, this allows processes and implementations to vary independently. We also showed how to enable the processes and the implementations for this scenario, both in the application and system areas, by using existing technologies and standards.

On demand SOA is about the fusion of business and IT. It is also about the integration of multiple technologies, which when taken in isolation provide some value, but which when used holistically provide far more significant payback to the enterprise. Thus, on demand SOA enables on demand business to reach a higher level of IT effectiveness.

Ultimately, this paper is about making on demand business a reality. SOA, itself, is the glue that binds business processes, applications, and IT infrastructure integration into a comprehensive architecture. On demand SOA takes this concept a step further with the fusion of business process transformation, service-oriented enabling technologies, and policy-based management.

ACKNOWLEDGMENTS

We are grateful to Rhonda Childress, Daron Green, Mark Ernest, Cary Perkins, and Chris Reech for their participation in the initial brainstorming sessions that were the basis for this paper. We are especially appreciative to Paul Magnone for posing this problem to us in the first place and to Scott Penberthy for providing financial support for the work summarized in this paper. C.H.C. is grateful for the constant management support of Nagui Halim, Dan Dias, and Asit Dan and for funding support from Dave Turek.

CITED REFERENCES

1. Web Services, World Wide Web Consortium, http://www.w3.org/2002/ws/.
2. A. G. Ganek and T. A. Corbi, "The Dawning of the Autonomic Computing Era," *IBM Systems Journal* **42,** No. 1, 5–18 (2003).
3. J. W. Ross and G. Westerman, "Preparing for Utility Computing: The Role of IT Architecture and Relationship Management," *IBM Systems Journal* **43,** No. 1, 5–19 (2004).
4. *The Grid 2: Blueprint for a New Computing Infrastructure (2nd Edition),* I. Foster and C. Kesselman, Editors, Morgan Kaufmann Publishing, Inc., San Francisco, CA (2003).
5. J. Hagel III, *Out of the Box, Strategies for Achieving Profits Today and Growth Tomorrow through Web Services,* Harvard Business School Press, Boston, MA (2002), p. 15.
6. K. Holley, *Powerful Enterprise Architecture and Information Technology Strategies, Why SOA is the Missing Link,* IBM Corporation (June 4, 2004), http://www.sys-con.com/story/?storyid=45100&DE=1.
7. H. Ludwig, A. Keller, A. Dan, and R. King, "A Service Level Agreement for Dynamic Electronic Services," *Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02),* Newport Beach, CA, IEEE Computer Society, Washington, DC (2002), pp. 25–32.
8. B. Carpenter and P. Janson, "Abstract Inter-Domain Security Assertions: A Basis for Extra-Grid Virtual Organizations," *IBM Systems Journal* **43,** No. 4, 689–704 (2004).
9. IT Infrastructure Library (ITIL), UK Office of Government Commerce, http://www.ogc.gov.uk/index.asp?id = 2261.
10. WebSphere, IBM Corporation, http://www.ibm.com/websphere.
11. F. Leymann and D. Roller, *Business Processes in a Web Services World—A Quick Overview of BPEL4WS,* IBM Corporation (August 2002), http://www-106.ibm.com/developerworks/library/ws-bpelwp/.
12. J. Aman, C. K. Eilert, D. Emmes, P. Yocom, and D. Dillenberger, "Adaptive Algorithms for Managing a Distributed Data Processing Workload," *IBM Systems Journal* **36,** No. 2, 242–283 (1997).
13. BEA WebLogic Server, BEA Systems, Inc., http://www.bea.com/products/weblogic/server/index.shtml.

14. Tomcat, The Apache Software Foundation, http://jakarta.apache.org/tomcat/.
15. Globus Resource Allocation Manager (GRAM), The Globus Alliance, http://www-unix.globus.org/developer/resource-management.html.
16. GridServer, DataSynapse, http://www.datasynapse.com/solutions/gridserver.html.
17. R. Levy, J. Nagarajarao, G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef, "Performance Management for Cluster Based Web Services," *Proceedings of 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003),* Colorado Springs, CO, IEEE Communications Society, New York, NY (March 2003), pp. 247–261.
18. Platform Symphony, Platform Computing, Inc., http://www.platform.com/products/Symphony/.
19. Platform Community Scheduler Framework (CSF), Platform Computing, Inc., http://www.platform.com/products/Globus/.
20. Platform LSF, Platform Computing, Inc., http://www.platform.com/products/LSF/.
21. Condor, The Condor Project, University of Wisconsin-Madison (UW-Madison), http://www.cs.wisc.edu/condor/.
22. OpenPBS, Altair Engineering, Inc., http://www.openpbs.org/main.html.
23. IBM LoadLeveler, IBM Corporation, http://www-1.ibm.com/servers/eserver/pseries/library/sp_books/loadleveler.html.
24. IBM Parallel Environment for AIX, IBM Corporation, http://www-1.ibm.com/servers/eserver/pseries/library/sp_books/pe.html.
25. IBM Tivoli Intelligent Orchestrator, IBM Corporation, http://www-306.ibm.com/software/tivoli/products/intell-orch/.
26. A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web Services On Demand: WSLA-Driven Automated Management," *IBM Systems Journal* **43,** No. 1, 136–158 (2003).
27. SAS IT Resource Management, SAS Institute, Inc., http://support.sas.com/documentation/onlinedoc/itsv/.
28. Evident Enterprise, Evident Software, Inc., http://www.evidentsoftware.com/products/entfeatures.aspx.
29. Tivoli Decision Support for OS/390, IBM Corporation, http://www.306.ibm.com/software/tivoli/products/tds-390/.
30. Unicenter NeuMICS Resource Management Analyzer Option for MQSeries, Computer Associates International, Inc., http://www3.ca.com/Solutions/ProductOption.asp?ID=1404.
31. HP OpenView, Hewlett-Packard Company, http://www.openview.hp.com/.
32. PATROL Enterprise Manager, BMC Software, http://www.bmc.com/products/proddocview/0,2832,19052_19429_23191_7266,00.html.
33. SAS IT Charge Management, SAS Institute, Inc., http://www.sas.com/solutions/itcharge/index.html.
34. CIMS, CIMS Lab, Inc., http://www.cimslab.com/default.htm.
35. Veritas Micromeasure, Veritas Software Corporation, http://www.veritas.com/news/press/PressReleaseDetail.jhtml;vrtsid=FV02LFBCMLLEHQFIYCLCFEY?NewsId=62001.
36. Singl.eView Billing and Commerce Software, ADC, http://www.adc.com/software/billingandcommerce/.
37. F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," *Proceedings of the Conference on File and Storage Technologies (FAST'02),* Monterey, CA (January 2002), pp. 231–244.
38. C.H. Crawford and K. Gildea, "Building a Distributed Shared Object Cache on Top of a Parallel File System," IBM Corporation, unpublished work.
39. DB2 Information Integrator, IBM Corporation, http://www-306.ibm.com/software/data/integration/db2ii/.
40. QoSWorks, Converged Access, Inc., http://www.convergedaccess.com/products/QoSWorks.html.
41. *Packeteer Application Traffic Management,* Packeteer, Inc., http://www.packeteer.com/resources/prod-sol/PacketeerBroFinal2.pdf.
42. M. Gupta, A. Neogi, M. Agarwal, and G. Kar, "Discovering Dynamic Dependencies in Enterprise Environments for Problem Determination," *Proceedings of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003),* Heidelberg, Germany, October 20–22, 2003, *Lecture Notes on Computer Science,* Vol. 2867, Springer-Verlag, New York (2003), pp. 221–233.
43. C. H. Crawford and A. Dan, "eModel: Addressing the Need for a Flexible Modeling Framework in Autonomic Computing," *Proceedings of the 10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2002),* Fort Worth, TX, October 11–16, 2002, IEEE Computer Society, Washington, DC (2002), pp. 203–208.

Accepted for publication July 12, 2004.
Internet publication January 7, 2005.

Catherine H. Crawford
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (catcraw@us.ibm.com).
Dr. Crawford is a Senior Software Engineer in the IBM Research Division. She is a member of the Distributed Computing Department and the Exploratory Stream Processing group. Her current research focuses on high performance and distributed computing. She is the author of several papers in the areas of distributed computing, numerical simulation, and wall-bounded turbulence. She received a bachelor's degree from the Massachusetts Institute of Technology (MIT) and master's and Ph.D. degrees in mechanical and aerospace engineering from Princeton University.

G. Paul Bate
IBM Global Services, 6710 Rockledge Drive, Bethesda, MD 20817 (batep@us.ibm.com). Paul Bate is a Certified Executive I/T Architect and Senior Technical Staff Member of the leadership team in the Enterprise Architecture & Technology service area within Application Innovation Services in the U.S.A. With over 30 years of experience in various roles within the IT industry, Paul is currently specializing in the emerging areas of grid computing, service-oriented architecture, and on demand computing. He is a technologist with deep expertise in enterprise technology architectures, leveraging reusable assets, and IBM best practices. Paul's contributions to IBM include the Adaptive Blueprint for Enterprise Architecture and commercialization of the ESS (Enterprise Solution Structure) e-business Reference

Architecture. Paul joined IBM in 1975 after receiving his Bachelor of Technology degree in computer science from the University of Bradford, England.

Luba Cherbakov
IBM Global Services, 6710 Rockledge Drive, Bethesda, MD 20817 (lubacher@us.ibm.com). Luba Cherbakov is an IBM Distinguished Engineer and a key technical leader of Application Innovation Services in the IBM Global Services organization. She is a recognized expert in enterprise architecture design and the development of complex distributed applications using component-based methods. She is also a key contributor to winning and delivery of many complex, first-of-a-kind IBM Global Services engagements across a wide variety of customer industries, with recent focus on the emerging areas of grid computing and service-oriented architecture. She is an author, advocate, and contributor to IBM's Reference Architecture assets, Architectural Description Standard, Adaptive Blueprint for Enterprise Architecture, and grid-computing offerings. She holds B.S. and M.S. degrees in computer science, with a major in software and systems and a minor in artificial intelligence and simulations.

Kerrie L. Holley
IBM Global Service, 425 Market Street, San Francisco, CA 94105 (klholley@us.ibm.com). Kerrie Holley is the CTO for IBM's SOA and Web Services Center of Excellence, and he is also the Chief Architect in Application Innovation Services (AIS). He is an IBM Distinguished Engineer and a member of IBM's Academy of Technology. He has 25 years of experience translating business requirements into process designs for cutting-edge network-centric distributed solutions. His responsibilities include technical oversight for network-centric projects, adaptive enterprise architecture design, IT strategy, formation of partnerships among clients and vendors, leading of architecture reviews, and management of technical risk. He has experience working in a variety of industries and has frequently advised senior management on how technology, Web-based technologies, software best practices, high-performance teams, and effective implementation can be used for competitive advantage within their businesses. Mr. Holley has a B.A. degree in mathematics from DePaul University and a Juris Doctorate degree from DePaul School of Law.

Charles Tsocanos
1608B Gerome Ave, Fort Lee, NJ, 07024 (chucklezt@yahoo.com). Charles Tsocanos received a B.S. degree from Rutgers University in electrical engineering and will receive an M.S. degree in telecommunications from Pace University. During his 9-year tenure at IBM he was involved in many leadership activities, including the development of consulting services and go-to-market strategies. He has extensive customer experience in business transformation and operational efficiency and has been a key contributor to IBM's on demand infrastructure services. Chuck is currently North American Director of Consulting for Equant Professional Services. ■