Dissecting Service-Oriented Architectures

By Boris Lublinsky and Dmitry Tyomkin

Numerous companies have experimented with their Web services pilot projects and found that this technology can be a viable addition to their development practices. The next step is to move beyond simple point-to-point Web services-based communications to broader application of these technologies. This requires an architectural shift to loosely coupled, standards-based Service-Oriented Architectures (SOA). Such architectures represent a new approach that requires a different perspective on the role of IT in the organization.

This article attempts to define SOA, where it fits in an organization, and how it relates to existing architectures. It describes how to quickly implement SOA.

business integration journal **takeaways**

BUSINESS

SOA:

- Increases business-user control over adjustment of core processes
- Enables better tracking and visibility into processes (also known as Business Activity Monitoring)
- Streamlines process execution and eliminates manual execution inefficiencies via enterprisewide processes.

TECHNOLOGY

- Contains complexity by providing a natural way to decompose complex problems into simpler, self-contained services
- Provides remote access to logic as needed rather than requiring its installation on all computers that might need it.
- Encapsulates data, allowing for clean separation between service interfaces (enterprise data dictionary) and the internal data of a service.

What Is SOA?

SOA is typically equated to Web services and is described using the W3C SOA architecture diagram (see Figure 1). While correctly depicting major elements of the Web services environment, this definition doesn't really help uncover the impact of SOA.

To understand this impact, start by defining Enterprise Architecture (EA). It's defined in the standards as, "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." The Open Group Architecture Framework (TOGAF) introduces an architecture description as "a formal description of an information system, organized in a way that supports reasoning about the structural properties of the system. It defines the components or building blocks that make up the overall information system, and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. EA is a conceptual tool that assists organizations with the understanding of their own structure and the way they work. It provides a map of the enterprise and is a route planner for business and technology change."

Normally, EA takes the form of a comprehensive set of cohesive models that describe the structure and functions of an enterprise. Important uses of it lie in systematic IT planning and architecting, and enhanced decision-making. The individual models in an EA are arranged in a logical manner, with an ever-increasing level of detail about the enterprise, including its:

- Objectives and goals
- Processes and organization
- Systems and data
- Technologies used.

The business, application, information, and technology perspectives of EA (see Figure 2) are important and commonly used.

The technology perspective lays out hardware and software supporting the organization. The application perspective, based on the technology perspective, defines the enterprise's application portfolio and is application-centered. The business perspective describes how business works, is based on the application perspective, and includes broad business strategies and plans for moving the organization to an envisioned future state. The information perspective traverses the other three and describes data required to properly manage processes, operations, and infrastructure.

SOA is an architectural style that promotes business process orchestration of enterprise-level business services.

SOA is an architectural style that promotes business process orchestration of enterprise-level business services (see Figure 3). The three major elements of SOA are:

- **Services**: SOA models the enterprise as a collection of business services, which are accessible across the enterprise. Monolithic stovepipe applications are dissolved in favor of self-contained business services that perform specific functions. These services can be invoked using a standard protocol, ensuring their availability across the enterprise and beyond.
- **Processes**: Business processes orchestrate the execution of these enterprise services to fulfill required business functionality. Enterprisewide processes essentially define an enterprise's operations.
- **Organization**: Organization owns all the SOA artifacts and governs their creation, usage, access, and maintenance.

SOA has a profound impact on all the perspectives of the EA, defined in Figure 2. Let's now consider in detail two major components of SOA: services and processes.

Services

Software development was always driven by the desire to minimize cost and increase flexibility. This creates requirements for:

- Software reuse
- Loose coupling
- Ease of integration.

Attempts to find this holy grail of software development have resulted in constantly changing mainstream application development practices — separating code into functions, Remote Procedure Calls (RPCs) for the transparent code distribution, object-oriented development for encapsulation, component-

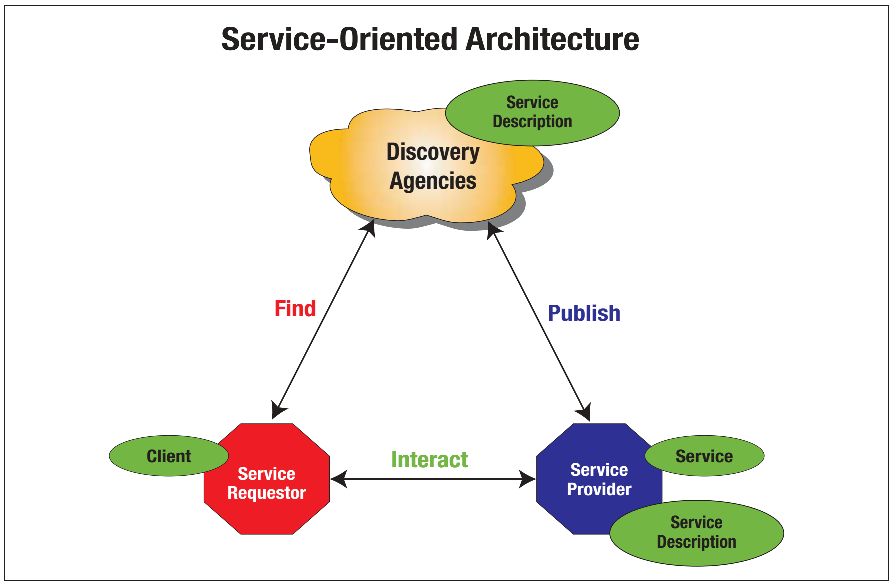

Figure 1 — SOA Definition From W3C

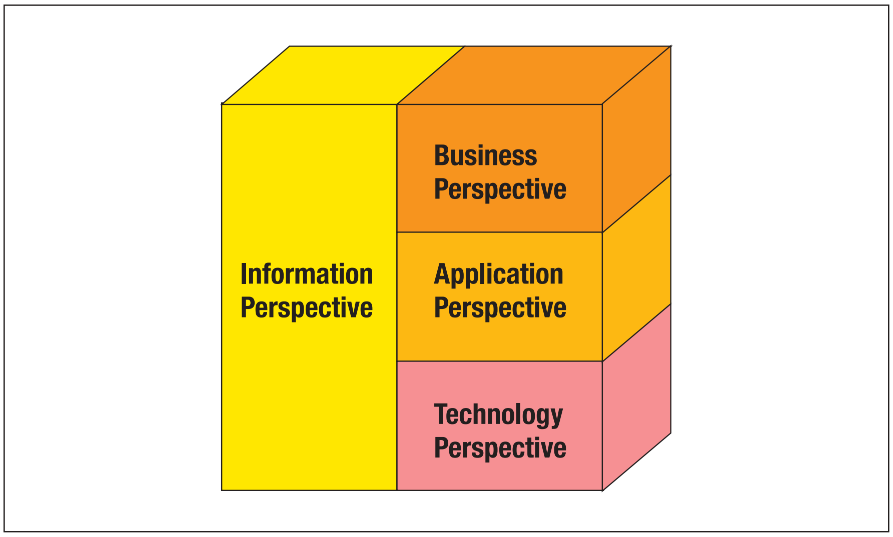

Figure 2 — Views of Enterprise Architecture

based development for standardization of software distribution and deployment, and now service-oriented development.

Service-oriented development is based on the concept of services. It's the realization of business functionality via software that anyone can use, anywhere, to compose new business applications by using these services in the context of new or modified processes.

This approach increases the reach of applications and enables continual software delivery. Service-based development allows for convergence of tightly coupled, highly productive aspects of n-tier computing (service implementation) with the loosely coupled, message-oriented concepts of messaging (service interactions).

A service is defined in terms of its interface, which is the way the service is exposed to the outside world. This includes the set of parameters (defining data required for interaction with the service) and communication protocol used for data transfer and actual service invocation. The service interface is defined by a service name and set of methods the service supports. Grouping of the methods in the service interface is defined by business functionality (requirements) of the service.

The following characteristics are typical for services:

- Business-driven
- Coarse-grained
- Process-centric
- Stateless invocation
- Loosely coupled
- Distributed
- Standards-based.

Normally, services provide both the business logic and state management relevant to the functionality they're designed to support. When designing services, the goal is to effectively encapsulate the logic and data associated with real-world processes, which is similar to Object-Oriented (OO) encapsulation. Because you can call services across a network, they should be coarse-grained. That is, services should wrap a substantial body of application logic, delivering value that justifies the latency cost of a network request. Similarly, services should expose coarse-grained interfaces. Rather than expose many interfaces that each manipulate small amounts of state, services should expose fewer interfaces that allow a single request to perform a complete function.

In summary, services provide a model for software design with built-in potential for integration and evolution.

Business Processes

Today's IT systems are mostly record-keeping systems. In the last 30 years, IT has captured terabytes of data, which are locked in stovepipe applications and numerous (often proprietary) databases. The methods, techniques, and mindset of IT today is to remain fixated on data capture, storage, and retrieval.

Businesses are defined through dynamic processes that are constantly expanding, contracting, and changing with business activities. Because processes are so hard to formalize and automate, they've long been second-class citizens in IT. Only the most basic, back-office processes are incorporated in most IT systems. By contrast, business processes of all shapes and sizes are the focus of management attention.

Under the data-centric IT paradigm, business people cannot obtain the information needed to compete on cost, quality, speed, and service. They need actionable information and knowledge. Companies that want to increase their effectiveness and competitiveness must make the process, not data or application, the basic unit of computer-based automation and support. They must shift their focus from records to processes. Process processing must replace data processing.

A business process is a set of activities carried out in a sequence to realize an objective. Every step contains associated actions, which are performed during that step, and business rules that define the transition to the next step.

According to the Workflow Management Coalition Terminology and Glos-

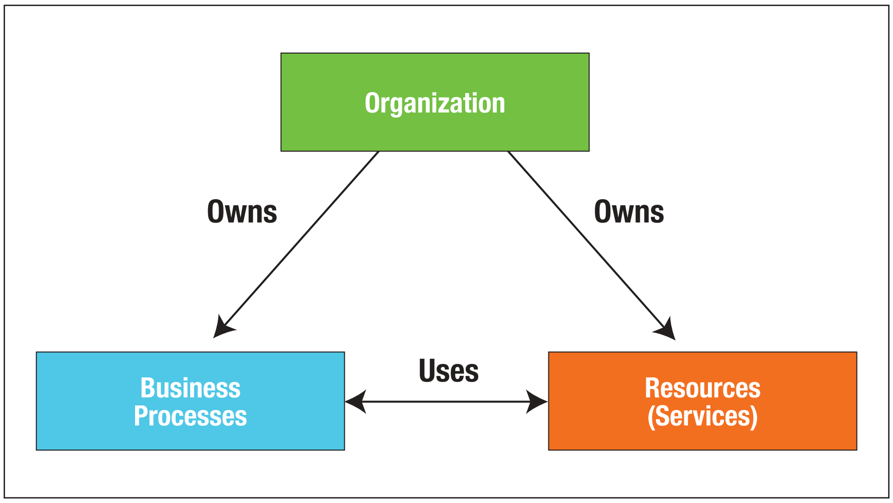

Figure 3 — Basic SOA Model

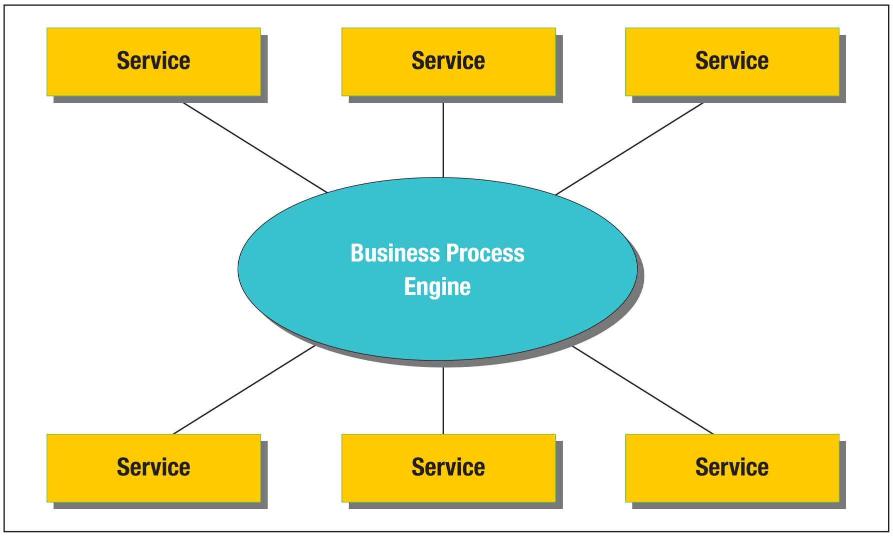

Figure 4 — Business Process-Driven Enterprise Architecture

sary, a business process:

- Is typically associated with objectives and business relationships such as an insurance claims process
- Has defined triggering (initiation) conditions for each new process instance (e.g., the arrival of a claim) and defined outputs at its completion
- May involve formal or relatively informal interactions between participants; its duration may vary widely
- Processes can be used in a higher-level business process as sub-processes.

Enterprisewide processes constantly evolve — processes also interact with other processes. They change at a rate that makes it difficult for applications and developers to keep up. The only feasible solution is to:

- Externalize processes, separating them from applications and providing tools to simplify process design, implementation, and changes.
- Design applications in the form of services that can play a role in the end-to-end process and let them access external data the process needs.

With this approach, the business process can interact with and coordinate services' behavior with respect to the overall goal of the process and each other. This shifts the focus from applications to processes.

Solving these process-related problems on a case-by-case, application-by-application basis does nothing more than create more islands of automation and record-keeping. The solution is to shift perspective to the end-to-end process, making it a centerpiece of EA, as shown in Figure 4. Transition to this architecture requires rethinking the applications, from being all-encompassing entities to services, integrated and orchestrated by the business process to support its required functionality.

SOA's Impact on EA

Again, SOA impacts all four perspectives of EA.

Business Perspective

SOA helps decompose functionality into more manageable, reusable parts, which can be separately designed, developed, and maintained. Adopting SOA means definition of enterprise functionality occurs in terms of services and processes. This approach allows for better alignment of business and application perspectives. If it's easier to trace the application perspective back to the business perspective, it's simpler to implement required changes in functionality. Such a strategy also facilitates separation of business services (fairly stable processing units) from processes (fast-changing elements of a business model).

Aligning functionality with underlying applications is also a more straightforward introduction of Business Activity Monitoring (BAM), the real-time monitoring of business events and transactions. Advances in business process implementation allow for tighter integration of BAM technologies with the execution of a business process. BAM delivers real-time business intelligence about the integrated enterprise to management.

Enterprise business process requirements drive discovery and definition of business services. Figure 5 provides a high-level, simplified algorithm for this.

The first step is to create a model of the business that identifies the fundamental entities and processes the business offers. Domain and business requirements should drive this. Part of this procedure involves identifying a set of services and interfaces to them for every process. As the model is completed and refined, the services are reorganized based on the common functionality of all required services. The next step is to map required services onto existing applications and define requirements for creating new services:

- If an application exists that supports required functionality, it must be validated, resulting in the specification of additional functionality that might be required. When this occurs, the functionality needs to be wrapped to provide interfaces that expose it for use outside the application.
- If there are no applications with the required functionality, then service interfaces must be solidified. Based on these interface definitions and requirements, the actual service implementation must be created. Creating new services doesn't always mean creating complete service functionality from

Enterprisewide processes constantly evolve — processes also interact with other processes.

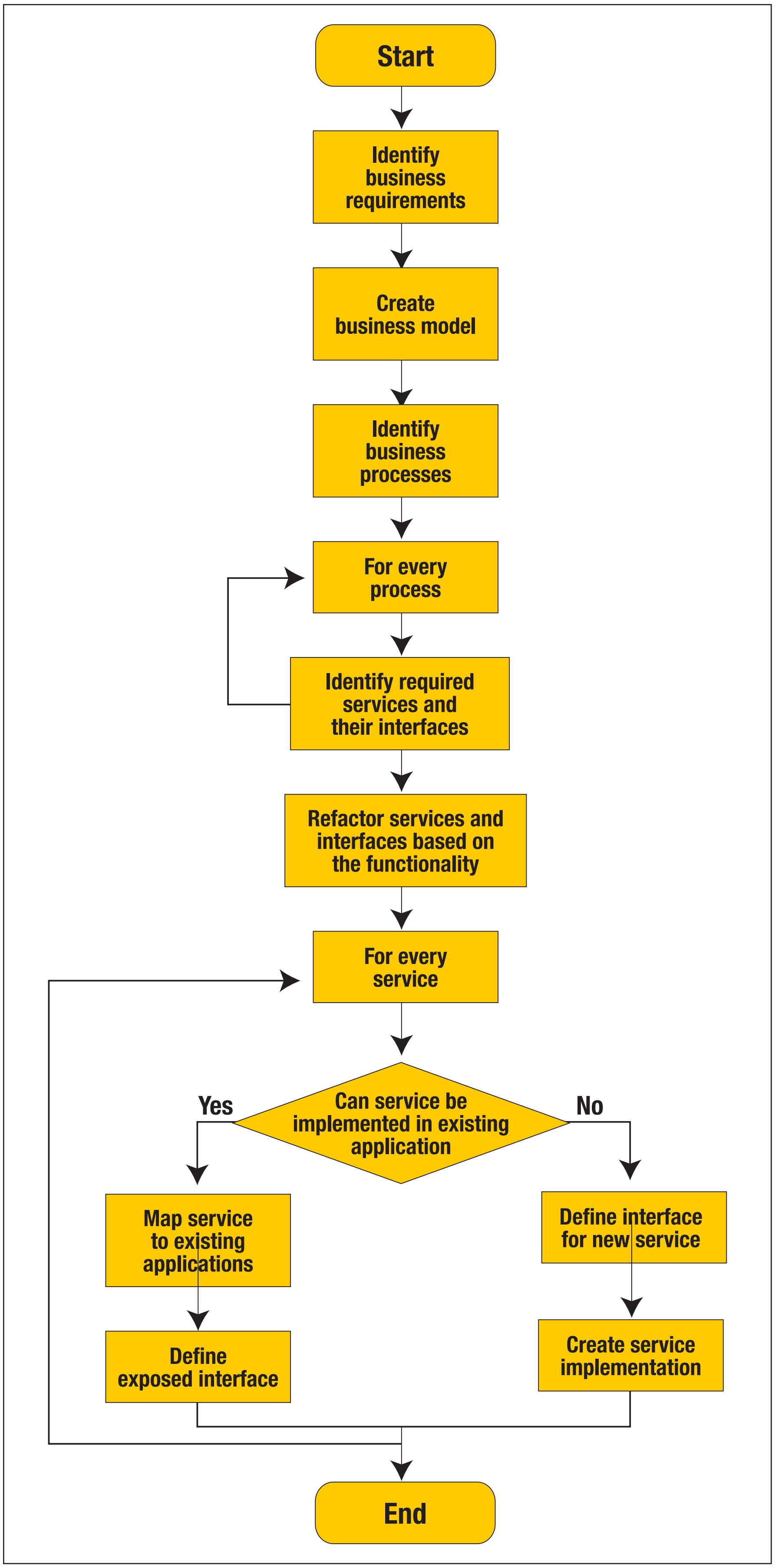
Start
Identify business requirements
Create business model
Identify business processes
For every process
Identify required services and their interfaces
Refactor services and interfaces based on the functionality
For every service
Can service be implemented in existing application
Yes
No
Map service to existing applications
Define exposed interface
Define interface for new service
Create service implementation
End

Figure 5 —Defining Business Services

scratch. Service implementation can partially delegate execution to one or more existing applications and supplement it with some new code.

Application Perspective

Introduction of services in the application perspective promotes loose coupling. Every service can be individually designed, constructed, and maintained. This approach eases overall maintenance of the application portfolio. Introduction of services leads to the application architecture shown in Figure 6. The architecture is comprised of disparate services communicating with one another over the service bus. Additional components are:

- Business process engine, allowing for externalizing the business process. Introduction of the separate business process engine provides separation of business process definition and execution from services implementation, allowing for even looser coupling in the application architecture.
- Service locator, allowing for externalization of the service location, supporting service location transparency
- Business services encapsulating the actual business functionality
- Utility services, which are special kinds of business services that don't belong to a firm's core business but can still be accessed by any client
- Common (infrastructure) services, providing system and infrastructure support for business services.

Compared to writing stand-alone applications, the process of designing, developing, deploying, managing, and maintaining robust services entails higher upfront costs.

Information perspective

The SOA implementation introduces two data-related concerns:

- Underlying data models for the service implementation
- Data dictionary for service messages, defining communications semantics of the SOA.

This separation of information perspective substantially simplifies the overall enterprise information strategy. In an SOA implementation, the data model exposed by the enterprise appli-

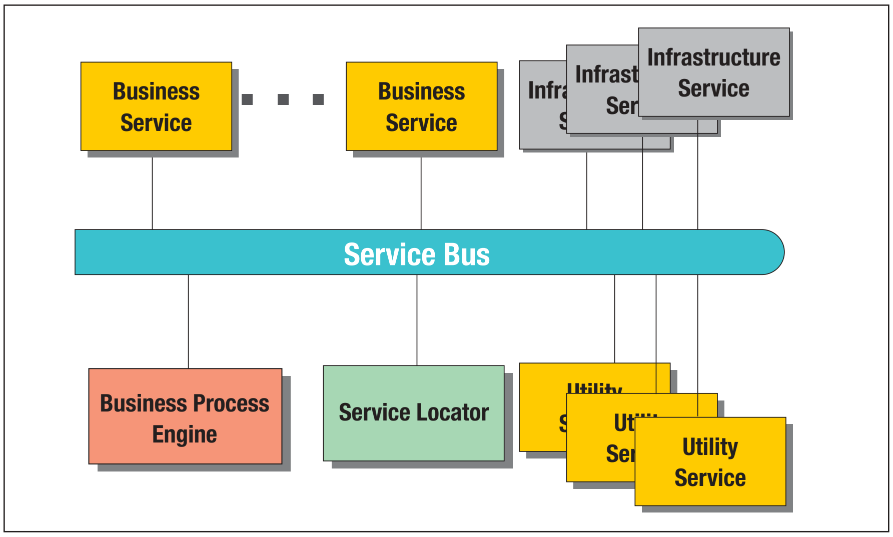

Figure 6 —Typical SOA Application Architecture

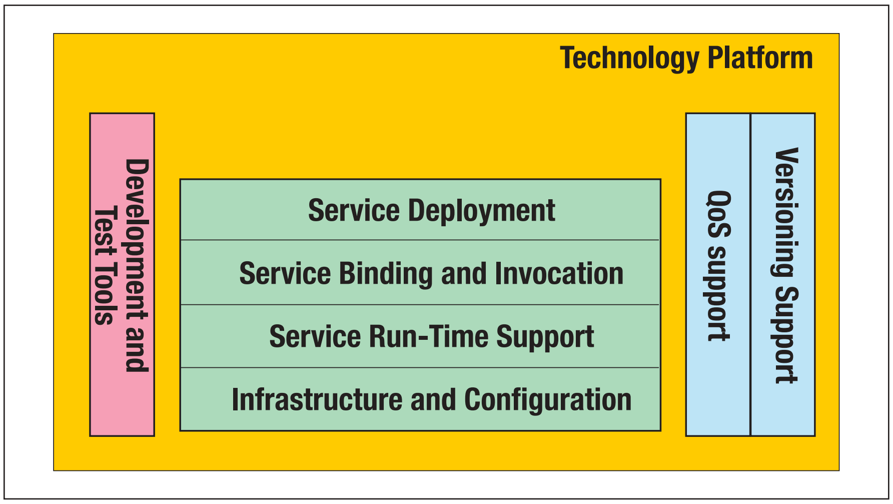

Figure 7 — Conceptual View of Technology Architecture

cation portfolio and used by internal and external enterprise processes is defined by the messaging data dictionary, which is decoupled from the underlying data model. Different services can use completely unrelated internal data models (they're encapsulated by services) as long as the data semantics that they're exposing adhere to the semantic data dictionary the enterprise adopts.

Because the messaging data dictionary is usually significantly simpler then the underlying data model, achieving common messaging semantics is usually easier than unifying the underlying data model.

Technology Perspective

It is a basis for building enterprise-scale services. It provides the foundation for the consistent policy and management of services development and maintenance. SOA usually requires developing an infrastructure that can be shared by many services delivering against diverse functional requirements. Operational support capabilities should be implemented once for all services in the organization.

Figure 7 shows a set of generic facilities (based on the service life cycle requirements above) that provide enterprisewide services for the SOA implementation. These facilities provide the common support required by any service in the application portfolio. Figure 7 contains three major parts, dedicated to the three technology platform components: tooling, infrastructure, and run-time support. The tooling component provides tools, processes, methodologies, and patterns required to design, develop, assemble, and test services and service-based processes. The infrastructure component provides basic infrastructure support for the services run-time, which consists of:

- **Service deployment**: concerns the processes and technology choices around deployment of services, including host platform
- **Infrastructure and configuration**: provides middleware, operating system, hardware, storage, networking, and the trust and management support for the whole system
- **Service run-time support**: hosts the process, logic, functions, and state management required by a service-based application and is the full enterprise application environment with specific support for services
- **Service binding and invocation**: contains services binding and invocation mechanisms, including support for both locating and invoking enterprise services and exposing applications or code as services in different operational environments.

The run-time component is responsible for the additional support for SOA and consists of two major parts:

- Quality of Service (QoS) support
- Service versioning support.

Achieving the Vision

Achieving the described levels of power and flexibility is difficult. It isn't feasible today to throw away existing IT systems and start with a clean slate. In reality, services are created by integrating and extending existing applications. You create them by aggregating components, some of which wrap existing applications.

As shown in Figure 8, typical service implementation exposes existing legacy

While useful, Web services standards are just a vehicle to help SOA delivery.

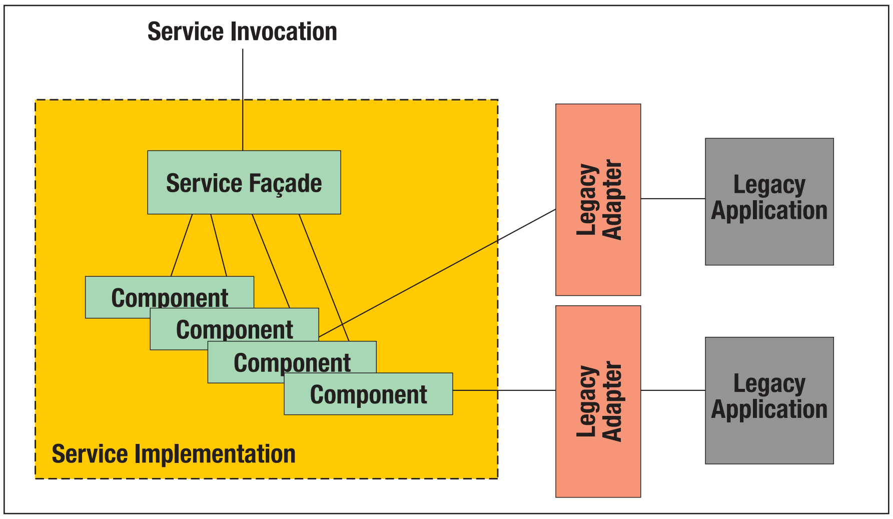

Figure 8 — Service Implementation

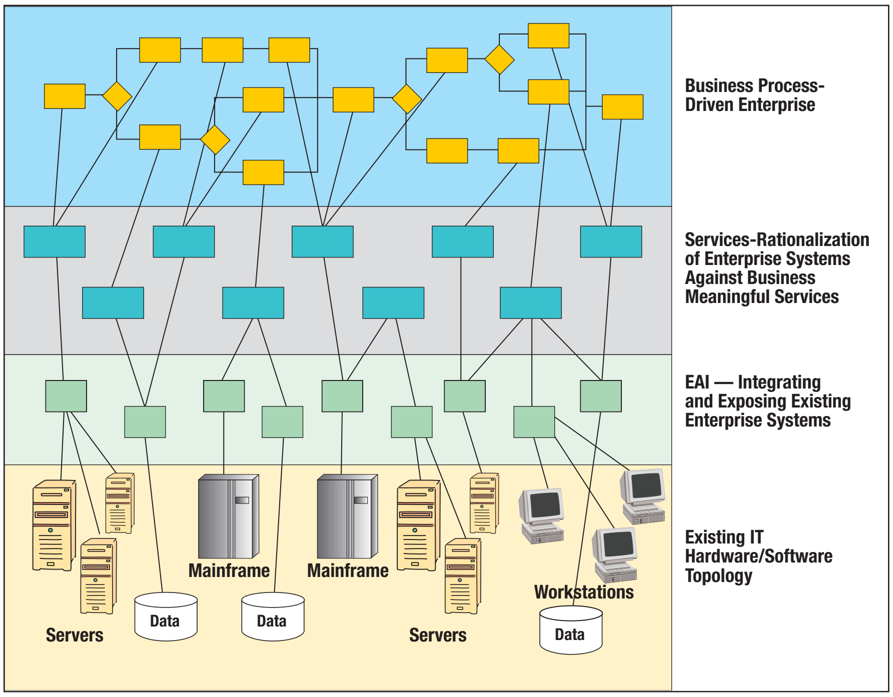
Business Process-Driven Enterprise
Services-Rationalization of Enterprise Systems Against Business Meaningful Services
EAI — Integrating and Exposing Existing Enterprise Systems
Existing IT Hardware/Software Topology
Mainframe
Mainframe
Workstations
Servers
Servers
Data
Data
Data

Figure 9 — Service Layer in the Overall Architecture

system functionality. It doesn't do it directly (as an EAI implementation would), but encapsulates the functionality in its own implementation, which allows for:

- Extending the legacy system functionality without touching existing legacy systems
- Increasing granularity of the service by combining the functionality of multiple legacy systems (or multiple interfaces of the same legacy system) and implementing additional functionality to rationalize this data
- No exposure of the existing application's portfolio.

The SOA approach differs from and is complementary to existing EAI initiatives. Most EAI projects are driven by IT and aimed at connectivity of the enterprise application portfolio. SOA enables the rationalization of existing applications against services, thus facilitating the convergence of IT with the business. This approach allows for building a working blueprint of the EA that supports business initiatives without a major overhaul of existing enterprise applications.

Additional driving forces for creation of this service layer include business process requirements. Although it's possible to create enterprise processes through EAI, based on the existing applications, this approach leads to embedding the existing application portfolio in the high-level enterprise processes. Introducing the services layer described above allows for creating a meaningful business abstraction layer between existing applications and processes. Figure 9 shows this architecture.

Conclusion

Organizations should use SOA to bridge the gap between diverse applications and address the growing demand for the power and flexibility of Business Process Management (BPM). The services layer, introduced by SOA, allows for direct mapping of business artifacts into an existing application portfolio. If properly implemented, SOA provides significant benefits for both business and IT. All four architectural perspectives (business, application, information, and technology) need to be addressed to deliver SOA. While useful, Web services standards are just a vehicle to help SOA delivery. BIJ

About the Authors

Boris Lublinsky is an enterprise architect at CNA Insurance, where he is involved in design and implementation of CNA's integration strategy, building application frameworks, and implementing service-oriented architectures. Prior to this he was a director of technology at Inventa Technologies, where he was overseeing and actively participating in engagements in EAI and B2B integration implementations and development of large-scale Web applications. e-Mail: boris.lublinsky@cna.com.

Dmitry Tyomkin is an enterprise architect at CNA Insurance. He has more than 15 years of industry experience. His career spans such diverse computing areas as mainframe programming, embedded systems, and Web development. He has worked in various industries, including telecommunications, retail, finance, pharmaceuticals, and software. Currently, he specializes in J2EE technology, XML-based communications, portals, and wireless computing. e-Mail: dmitry.tyomkin@cna.com.