zapthink white paper

SERVICE-ORIENTED DATA ACCESS

BUILDING INTEROPERABLE, ROBUST & REUSABLE DATA SERVICES

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com

SERVICE-ORIENTED DATA ACCESS

BUILDING INTEROPERABLE, ROBUST & REUSABLE DATA SERVICES

September 2007

Analyst: Jason Bloomberg

Abstract

Service-Oriented Architecture (SOA) is an approach to organizing IT resources and data to meet the changing needs of the business. Implementing SOA depends upon the IT organization being able to build interoperable, robust, reusable, and composable Services that abstract the underlying application functionality and data in the organization. To put this building block vision of SOA into practice requires a solid technical foundation, which includes a persistence layer that facilitates interaction with heterogeneous data sources that store and provide the structured and unstructured information that the enterprise runs upon.

The key to enabling SOA with such a persistence layer, in turn, depends upon abstracting access through data access technology. Technologies such as JDBC, ODBC, and ADO.NET play an integral role in the design and development of a SOA Data Services strategy. With best-of-breed data access technology in place, the organization stands a good chance of succeeding with their SOA efforts. If an organization drops the ball on data access, however, it's unlikely the Services will exhibit the key building block characteristics the organization needs to meet their agility requirements.

I. The Context of Service-Oriented Architecture

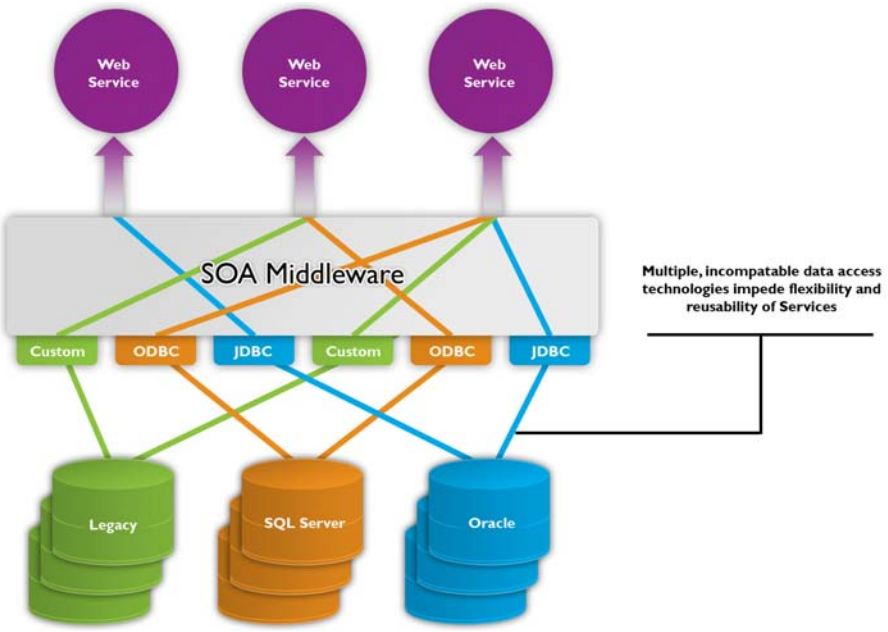
It's virtually impossible to be in an enterprise IT shop these days without having some involvement with Service-Oriented Architecture (SOA). It seems that every large organization has SOA on their roadmap somewhere, and furthermore, SOA efforts tend to grow like spider webs, eventually touching every corner of IT, as well as the business. Case in point: SOA efforts are now impacting the teams responsible for data persistence in the organization. Database architects and operators, information specialists, data integration experts and others are now being called upon to contribute to their enterprise's SOA initiatives.

The relationship between data and SOA is multifaceted.

As it turns out, the relationship between data and SOA is multifaceted. On the one hand, Services depend upon data persistence like all software. But on the other hand, SOA requires that companies address particular challenges that rely upon the data team's expertise to solve. After all, virtually all enterprises face the challenge of accessing information in the organization. Information locked away inside monolithic application silos has proven a stubborn obstacle to providing the flexibility the business requires. For organizations to be successful with SOA, therefore, they must solve the technical challenges of accessing data across their organization, if they have any hope of building flexible Services that offer the performance and agility the business requires.

It's also important to remember that while many organizations look to SOA for both increased agility as well as reusability of existing assets, poor choices in data access middleware can actually limit both agility and reusability. When architects fail to pay sufficient attention to data access issues and attempt to layer a Service oriented approach on top of their existing data sources, they often find that providing flexibility above the Service abstraction requires complex changes at the data source level, thus preventing the agility they require, as shown in the figure below.

The Problematic SOA Data Layer



Source: ZapThink

In the figure above, an organization has implemented Web Service interfaces as part of a SOA initiative, and yet, they haven't properly addressed the data access issues that underlie those Services. Instead, it's critical to address data access—the technology that enables applications, Service implementations, and other software to access heterogeneous data sources—by leveraging data access middleware that provides the performance, reliability, and flexibility necessary to support the agility and reuse benefits of SOA. The bottom line is that regardless of the abstraction and persistence technology that make up the SOA infrastructure, data access remains a key building block technology for SOA, for example, an organization using Hibernate to abstract data will suffer performance penalties if the underlying JDBC driver is not fast and scalable.

The role of data in SOA

Service-Oriented Architecture (SOA) is an approach to distributed computing that treats software resources as Services available on the network, where people can compose those Services into business processes to meet business requirements in a flexible, agile manner. Consumers of these Services (the software that accesses the Services) can find and connect to the desired Service providers (the software that offers the Services) in a *loosely coupled* fashion, meaning that the Service providers can be independently created, controlled, developed, and managed from the Service consumers.

From the business perspective, Services represent functionality and data the organization requires to run its processes and thus meet the needs of the business. Business people don't care if some capability consists of application functionality while another actually represents a data operation, and furthermore, it's also not relevant to the business whether data are structured or unstructured, or where those data are stored or how they get to the user. Business users simply want Services to work as advertised.

The data challenge for the enterprise as it implements SOA focuses on dealing with the various types and sources of data in the organization transparently to the business user.

The data challenge for the enterprise as it implements SOA, therefore, focuses on dealing with the various types and sources of data in the organization transparently to the business user. Implementing SOA depends upon exposing information and processes as self-contained Services that can communicate and interoperate with each other in a standard way, enabling the business to build flexible compositions of Services that implement business processes.

Addressing this SOA data challenge requires the appropriate use of Data Services. A *Data Service* is a contracted, composable representation of a data query or a combination of data queries. In essence, Data Services abstract both data sources as well as cross-database queries that result from EII or other data integration steps. Abstracting the underlying data infrastructure provides the separation between data and implementation that is essential to providing data building blocks that organizations can compose into reusable, composable business Services.

Individual Data Services, however, only provide a limited benefit. To fully apply SOA best practices to data, it is essential to create a Data Services layer. In the

context of SOA, a *Data Services layer* consists of Data Services as well as the infrastructure necessary to deliver a production-quality runtime that manages delivery and persistence of data among Service consumers, business Services, and the multiple data sources such Services use, whether in an enterprise or in a cross-enterprise distributed computing environment.

The core strengths of business Services

SOA presents business functionality and data as abstracted Services.

In essence, then, properly architected SOA presents business functionality and data as abstracted Services. To understand the business context for such Services, it's important to highlight four strengths of business Services in the context of SOA:

- *Services are interoperable* – Standards-based interfaces are a key enabler of the interoperability the business wants from their Services.
- *Services are robust* – We want from SOA a design for robustness, as the business requires IT to function regardless of the business problem at hand.
- *Services are composable* –To implement flexible business processes, the business wishes to compose Services into applications that implement business processes in a flexible way.
- *Services are reusable* – For example, if an organization has a set of core customer information Services, they should be able to leverage them in many different business processes.

There's little the business wants from IT that doesn't fall under the four characteristics above. The business simply wants composable, reusable, interoperable, robust capabilities they can leverage in flexible ways to meet the changing needs of the business. It is the responsibility of IT, however, to live up to these requirements for Services. To achieve these four characteristics, you want to follow technical best practices across the board. For example, reusability is an important characteristic, but if the code underlying a Service is of poor quality, then reusing it may not be desirable. Furthermore, as we'll see below, when you build Data Services, it's important to use best of breed data access technologies. To this end, it is critical to choose interoperable and highly scalable data access middleware such as ODBC and JDBC drivers and ADO.Net data providers.

In fact, reuse is a core SOA benefit, yet code reuse, including data access code, has always presented many challenges. The way that SOA addresses the challenge of reuse is by considering reuse of Services at runtime. Instead of code reuse, Service reuse leverages the loose coupling of the Services to enable those Services to be broadly consumable by different people in different contexts. It is the challenge of IT, therefore, to adequately architect the infrastructure for maintaining the loose coupling of the Services.

Service reuse leverages the loose coupling of the Services to enable those Services to be broadly consumable by different people in different contexts.

The shift to supporting reuse with well-architected infrastructure is a subtle, but profoundly important part of understanding how SOA works. For example, consider data access code. In traditional distributed architectures, developers may choose to write a data access object, which they might then seek to make reusable for different purposes. However, if there's a problem with that data access code, then reusing it simply compounds the problem.

Furthermore, whenever anything changes, including the underlying database, the data model, or the version of their coding environment, for instance, they need to update that data access code everywhere it appears—a clear example of the

downside of tight coupling. Instead, if they abstract data access as Data Services, and move the data access code into supporting infrastructure, then they are able to support changes throughout the environment in a much more flexible manner: an instance of the power of loose coupling.

II. Data Integration and SOA

As organizations move forward with their SOA implementation plans, they often find themselves grappling with significant, long-lived problems of data and semantic integration among a wide variety of data sources. These data sources range from structured data stores, as in the case of relational databases, mainframe data sources, and enterprise applications, to semi- or unstructured data such as Web pages, PDF documents, Office files, XML documents, email, media content, print streams, or a wide variety of content and data feeds and formats. The need to access and process information of so many disparate types from so many disparate sources forms the data integration challenge that organizations must deal with today.

The challenge of data in the SOA context

Ask virtually any business executive about the core value IT provides to the business, and they are likely to respond that "data" (the information we intend to understand and use), rather than "applications" (the systems that process that information) are the lifeblood of any organization. In many ways, they are right; information forms the basic foundation and reason for being for IT. However, the two concepts of data and applications are inextricably linked. Data by themselves are often inaccessible and unintelligible without the applications that process them, and applications serve no usable purpose without data.

In the context of SOA, the abstraction of Services and the concept of loose coupling prevent business users from having to know if the data they are consuming originated in a database, an enterprise application, a file system, another company, or anywhere else for that matter. In fact, the data users consume is abstracted from the source of the data. This promise of ubiquitously accessible data freed from the constraints of their sources is liberating for companies as they struggle with integration challenges. However, since SOA abstracts the source of data, there is significant overlap among data integration technologies like Extract-Transform-Load (ETL) and Enterprise Information Integration (EII), data transport and messaging technologies like message buses, Enterprise Service Bus (ESB) and B2B integration products, and Enterprise Application Integration (EAI) suites.

The data challenge of SOA goes beyond traditional integration requirements.

Fundamentally, the data challenge of SOA goes beyond traditional integration requirements. Consuming data within SOA requires an abstracted approach to data access, data transport, and processing data with applications. Furthermore, significant heterogeneity in the data themselves as well as the technology that organizations use to access, move, and integrate those data redoubles the challenge of SOA. There's no question that architects who focus on SOA must take a hard look at solving the data access and integration challenges in order to plan a successful, loosely-coupled architecture.

Solving the data integration challenge

One key part of the SOA data challenge is enabling timely access to critical, business-relevant information. Enterprises must often cobble together information of many disparate types from many disparate sources forms in order to make sound business decisions. However, many times this information arrives

too late to actually enable the business to make those decisions in an agile fashion.

In addition, certain business operations, such as fulfilling customer purchases, billing operations, and real-time operations such as stock trading and online banking require instantaneous access to information distributed throughout the organization from a wide array of heterogeneous data sources. In order to meet their needs, companies have resorted to making redundant copies of information just so that it is available to those critical systems when people need them. The end result is a tangled web of redundant information, dispersed throughout the organization and inevitably out of synch. This situation becomes even more complex when integration with far flung business partners, suppliers, and government agencies becomes important.

Among the most important benefits of SOA is building more flexible approaches to data integration and data access.

Among the most important benefits of SOA is building more flexible approaches to data integration and data access. Companies are increasingly realizing the application integration benefits of SOA, but they are also finding that simply getting systems to talk to each other is a small part of the challenge of making distributed computing in a heterogeneous infrastructure work. Indeed, the bigger challenge comes from trying to piece together data from these multiple systems in a way that makes them easy to understand and process.

III. The Role of the Data Services Layer

While SOA focuses on the business Services that organizations compose to implement flexible business processes, the architectural approach is equally applicable to aspects of an organization's heterogeneous infrastructure. In fact, to address many of the data access and integration issues we've discussed in this paper, building Data Services is a key best practice.

The Data Services layer abstracts the data sources that are relevant to the business Services available in the organization. In essence, the Data Services layer provides a single abstracted point of access for all data access, update, and creation operations. The Data Services layer acts as a bridge between the business Services and the underlying data persistence layer. The Data Services layer also provides a holistic view of the data models that the underlying persistence layer relies upon.

Architecting and building a Data Services layer

A Data Services layer must provide an interface that exposes a standard set of reusable Data Services for reading and writing data independent from the underlying data sources. One of the important benefits of a Data Services layer is that it enables loose coupling between the applications using the Data Services and the underlying data source providers. Loose coupling enables database architects to modify, combine, relocate, or even remove underlying data sources from the Data Services layer without requiring changes to the interfaces that the Data Services expose. As a result, the database architects can retain control over the structure of their data while providing relevant information to the applications that need it. Over time, this increased flexibility eases the maintenance of enterprise applications.

Before the advent of SOA, developers built the capabilities that the abstracted Data Service layer would provide using manual coding, tightly embedding that code into the application under construction. Embedding such data access and data abstraction code directly into applications limits the flexibility and reusability of the resulting applications, and as a result, enterprises looked to traditional

middleware products like ETL and EAI to provide capabilities of the Data Service layer from a middleware perspective.

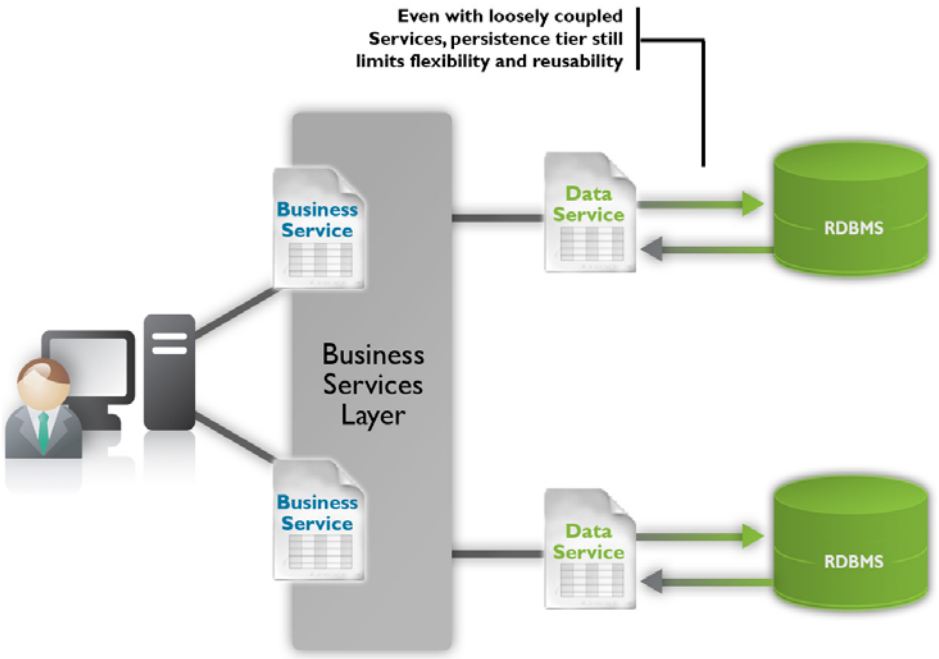
The ETL approach typically works well for small numbers of enterprise applications that are read-only or query-intensive and can tolerate infrequently refreshed data. However, changes such as adding new data sources typically require a partial redesign and reloading of the centralized data store, so the ETL approach is best suited for static applications that don't require the flexibility that SOA can provide. ETL can be costly as well, and requires a high management overhead.

EAI approaches are another common approach for building a tightly-coupled data interaction layer. With EAI, developers write workflows that include code for access, validation, transformation, and joining steps. As a result, developers must explicitly code details such as the order of access, the transformation logic, and join methods into each workflow. As a result, the EAI approach centralizes data interaction logic, but still does not provide the flexibility that many organizations require from their Data Services layer.

Even when an organization is leveraging Services in the context of SOA, a poorly architected Data Services layer can lead to performance issues.

Even when an organization is implementing SOA, however, a poorly architected Data Services layer can lead to performance issues. In many cases, the applications the organizations wish to abstract each have their own database which contains a duplicate copy of business reference data such as customer information, product information, and inventory levels, as shown in the figure below. The organization must synchronize such databases on a regular schedule, which can lead to stale or inconsistent data between applications. In this case, even when the organization has exposed application functionality as loosely coupled Services, the persistence tier still limits the flexibility and reusability of the Services.

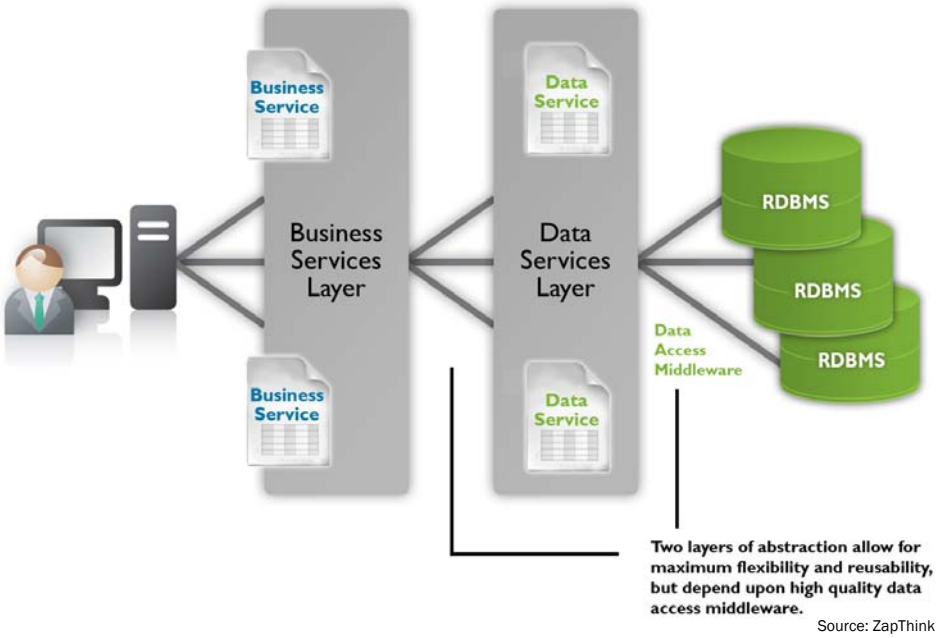
Loosely Coupled Services Abstracting Replicated Data



Source: ZapThink

Incorporating a Data Services layer into the SOA implementation addresses the problems above. Data Services offer transaction and connection management to multiple applications, as shown in the figure below. The Data Services layer manages the relationships between the Data Services and ensures that each application is aware of all data changes, regardless of the cause of the change. Leveraging Data Services as infrastructure Services beneath the business Service abstraction increases reusability and flexibility, and shortens development and rollout time for new Services. A well architected Data Services layer relies on consistent, high performance data access middleware that leverages industry standard APIs such as ODBC, JDBC, and ADO.NET, and SDO.

The Data Services Layer in a SOA Implementation



Source: ZapThink

As a result, building a Data Services layer as part of a SOA implementation provides the infrastructure necessary to reap the full benefits of SOA. Specifically, Data Services layer provides the following solutions to common data problems:

- *Reduces reliance on hand-coded persistence logic* – by building abstracted, shared Data Services, it's possible to leverage best-of-breed data access middleware to support the Data Services as part of an architected plan.
- *Provides a solution to inflexible integration* – instead of the point-to-point or hardcoded integrations of traditional integration approaches, the Data Services layer provides for loose coupling at the persistence tier.
- *Removes data query bottlenecks* – The Data Services layer enables organizations to implement content-based routing techniques to avoid bottlenecks.

- *Improves data consistency* -- The Data Services layer functions as a single locus for data access in the enterprise, helping an organization drive toward a single source of truth for its data. Implementing a Data Services layer helps ensure that applications always pull data from the correct sources, and consistently provide appropriate information back to all applications.

Addressing potential data access pitfalls

Accessing the data in the various data stores across the enterprise in the most efficient, flexible manner is the basis for building Data Services, and is thus critical for building all the Services in the SOA implementation.

It's important to realize that the foundation that the Data Services layer is data access. Data access, in turn, depends upon standards like ODBC, JDBC, ADO.NET, and SDO. Even when using a persistence layer like Hibernate, data access middleware is critical. A slow JDBC driver under Hibernate or a slow ADO.NET provider under Microsoft Linq will inevitably slow down the Services. In many ways, accessing the data in the various data stores across the enterprise in the most efficient, flexible manner is the basis for building Data Services, and is thus critical for building all the Services in the SOA implementation.

In essence, the Data Services layer virtualizes data access, which means that those Services can hide a plethora of potential data access pitfalls, including:

- Performance and scalability issues.
- Database platform, application, and version differences.
- Differences in how the various data sources handle the details of standard data access operations such as create, read, search, update, and delete.
- Data source connectivity issues arising from network troubles or other low-level difficulties.
- Data source security priorities, which might vary from database to database, table to table, or even row to row.
- Data mapping challenges arising from semantic differences among heterogeneous data.
- Problems arising from the mixture of structured and unstructured data, like file formatting issues.
- Differences in versions of SQL.
- Transaction integrity issues, especially across heterogeneous data sources.
- Exception handling and reporting.
- Low-performance data access, where a single trouble spot can act as a bottleneck, slowing down several Services.

Data access has long been a limiting factor in scalability, performance, and interoperability, but SOA magnifies this problem because organizations are both more likely to reuse the underlying code and also leverage the existing Data Services across many Services and composite applications. The first step to resolving these data-related issues is to build shared, centralized Data Services, which will result in an adaptable and maintainable SOA implementation.

In such a scenario, data access logic only appears in one place, no matter how many applications consume it. As a result, instead of scattering data-related Service invocation code throughout each business Service, centralized data

access helps to create an environment that enables a best-of-breed data access solution to address the above data access issues.

Data access as fundamental building block for SOA

Choosing best-of-breed data access middleware is an important enabler of SOA.

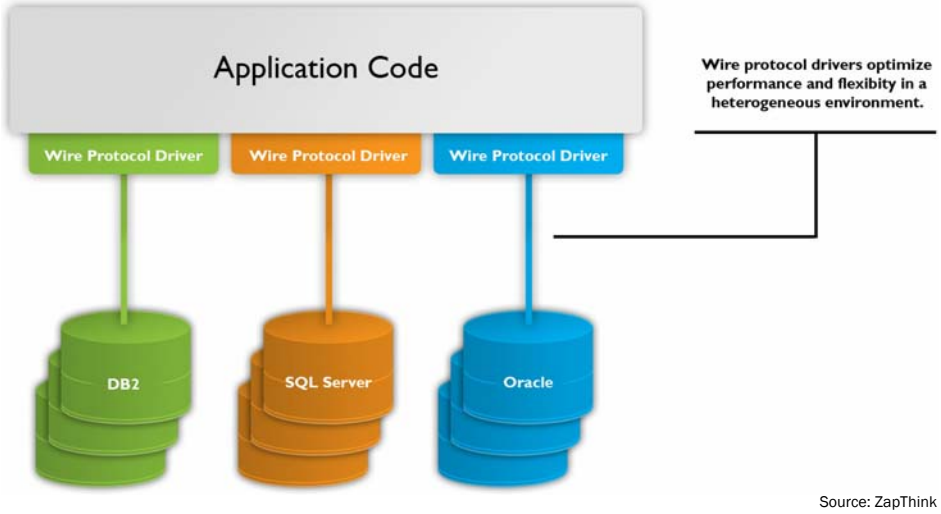
Utilizing the above Data Services approach results in a flexible structure with data access at its foundation. SOA depends upon loosely coupled business Services, which in turn depend upon a Data Services layer that relies upon Data Services that has data access as a fundamental building block. As a result, choosing best-of-breed data access middleware is an important enabler of SOA. Even if someone is just getting started with SOA, they should use the very best data access middleware to avoid problems down the road.

Data access middleware has always played an important part in the performance and scalability profile of an application. The Services context amplifies this importance, because system demands on infrastructure tend to be higher and reliability is every bit as critical as with traditional integration. Dynamic environments where multiple Services reuse data access code and new Services go into production regularly require that such code meets rigorous requirements.

Furthermore, choosing the appropriate data access middleware for the system the architect is designing will improve the overall performance, scalability, and flexibility of the system. To this end, architects should look for specific attributes for their data access middleware, including capabilities for boosting query performance, such as connection pooling. This middleware should also support tunable data access performance such as adjusting network packet size. For scalability and high availability, data access middleware should be multithreaded and thread safe, and offer client load balancing and failover to alternate servers.

It's also important for data access middleware to support different types and versions of databases as well as all the subtle variations in SQL they support. Such support for heterogeneity should also extend to multiple computing platforms, chipsets and operating systems. In addition, for the best performance and flexibility, data access middleware should offer wire protocol drivers to avoid the overhead and maintenance issues of off-the-shelf database drivers, as the figure below illustrates. Such drivers must support the full range of relevant standards, including JDBC, ODBC, ADO.NET, and over time, SDO.

Data Access Middleware Leveraging Wire Protocol Drivers



Source: ZapThink

Finally, companies should incorporate data access middleware in a comprehensive IT security strategy that covers both network security and database security, with secure communications and secure code. It should also integrate with multiple solutions for authentication and authorization.

Examples of Data Services layer in production

Here are two examples of how selecting best-of breed data access middleware helped organizations with their SOA initiatives:

- *Equities trading* – a large brokerage house had demanding requirements for reliability, performance, and scalability, processing thousands of transactions per second at peak volume. Via Data Services technology, they deliver immediate responses to their customers, and they're able to scale to meet the needs of their business with near-perfect uptime. As part of their SOA initiative, they implemented dozens of business Services that utilize the same Data Services. The consuming applications share a common data model as well as common data. They selected data access middleware that provided optimized updates, load balancing, and failover.
- *Logistics* – a logistics provider built their business on being able to deliver custom orders to locations around the world on short notice. The data they required to perform such scheduling consists of accessing large quantities of data, from customer preferences, to transport availability, to special handling instructions. Because they chose a Data Services infrastructure that allowed them to optimize performance, their architects found scalability to be nearly linear across all servers.

IV. DataDirect Technologies: Solving the Data Access Problem for SOA

DataDirect Technologies, an independent operating company of Progress Software Corporation, has a track record of success in the data access middleware space. Their middleware is in use by vendors of leading products for database management, business intelligence, integration services, application services and SOA, including a wide range of database drivers for major databases and other applications, including ODBC drivers, JDBC drivers and ADO.NET data providers. DataDirect's products are also meeting the most rigorous data access needs of large enterprises around the world.

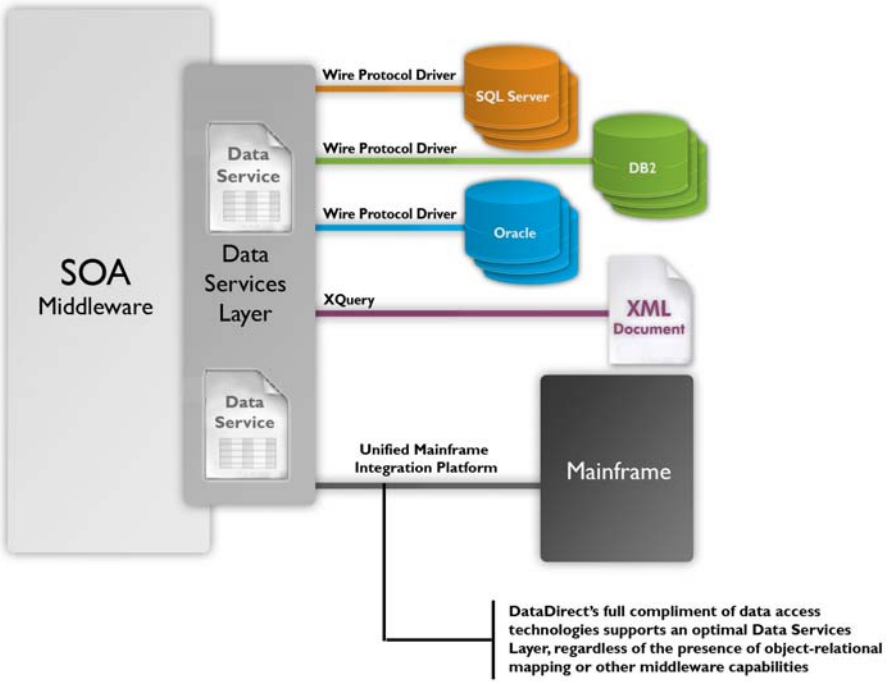
DataDirect's *Connect* database drivers have several differentiating characteristics that put them in a different category from freely available or open source drivers, as well as the drivers that the database vendors provide. In particular, the wire protocol design of DataDirect's drivers obviates the need for database client software and libraries, simplifying installation and administration as well as improving performance. In essence, the wire protocol design offers a single, efficient layer between Data Services and the data sources.

The wire protocol design of Data-Direct's drivers obviates the need for database client software and libraries, simplifying installation and administration as well as improving performance.

DataDirect's Connect drivers also provide standards-based data connectivity and optimized performance based on the direct communication between the Data Services layer and the wire protocol. The drivers also offer performance tuning via DataDirect's *Performance Tuning Wizard*, which ensures optimal performance out-of-the-box by enabling the administrator to tune the environment by answering a series of questions related to the target environment. Based on this information, the wizard automatically generates the connection properties necessary to configure the driver for optimal performance.

Customers leverage DataDirect's data access products to create an expanded data tier that is a critical enabler of a Data Services layer, as shown in the figure below.

The Expanded Data Tier



Source: ZapThink

Perhaps the key value propositions of the Connect drivers are performance, scalability, interoperability, and the ability to deal with multiple versions of a single database. The latter point is especially important, as Connect resolves database version issues, platform and operating system differences, and also provides for SQL leveling, which addresses the differences among SQL versions.

V. The ZapThink Take

Should the data access in the organization fail to perform, then the Services at the center of the architecture will also fail.

Fundamentally, SOA provides an Enterprise Architecture approach that spans the requirements of the enterprise: from the business context across the Services abstraction to the underlying infrastructure. Data access is an important part of that architecture, even though many architects haven't realized it yet. Architects will have to agree, however, that should the data access in the organization fail to perform, then the Services at the center of the architecture will also fail.

Businesses demand robust, interoperable, reusable, composable Services from IT. It's up to IT to leverage best practices to build Services that live up to that promise. As a result, it's important to choose best of breed data access middleware as a critical building block for any SOA initiative. Data access is a fundamental building block of SOA, and if organizations fail to make good choices there, the entire Services infrastructure will suffer. Fundamentally, regardless of the SOA infrastructure that runs above the Data Services layer, there's no question that data access remains a key building block technology for SOA.

DataDirect's Connect family of data access tools address the data access issues for many organizations, including those enterprises who are implementing SOA. DataDirect has established a sound value proposition that leverages the performance and flexibility of its products. Now that organizations are implementing SOA, it's important to connect the dots between building Services and data access.

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an IT advisory and analysis firm that provides trusted advice and critical insight into the architectural and organizational changes brought about by the movement Service Orientation and Enterprise Web 2.0. We provide our three target audiences of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink helps its customers in three ways: by helping companies understand IT products and services in the context of Service-Oriented Architecture (SOA) and the vision of Service Orientation, by providing guidance into emerging best practices for Web Services and SOA adoption, and by bringing together all our audiences into a network that provides business value and expertise to each member of the network.

ZapThink provides market intelligence to IT vendors and professional services firms that offer XML and Web Services-based products and services in order to help them understand their competitive landscape, plan their product roadmaps, and communicate their value proposition to their customers within the context of Service Orientation.

ZapThink provides guidance and expertise to professional services firms to help them grow and innovate their services as well as promote their capabilities to end-users and vendors looking to grow their businesses.

ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into the best practices for planning and implementing SOA, including how to assemble the available products and services into a coherent plan.

ZapThink's senior analysts are widely regarded as the "go to analysts" for SOA and Enterprise Web 2.0 by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry. ZapThink was founded in November 2000 and is headquartered in Baltimore, Maryland.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Road
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (815) 301 3171
info@zapthink.com

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com