Oracle Modernization

Oracle IT Modernization Series
Modernization: The Path to SOA

An Oracle White Paper
August 2008

ORACLE

Modernization: The Path to SOA

Introduction

More and more organizations are looking to service-oriented architecture (SOA) as the basis of their future computer architecture. Recognizing that legacy application design and implementation approaches have led to applications that are costly to operate and maintain, hard to change, and rely on a dwindling set of skills, organizations are hoping that SOA provides a key component of the answer to these problems.

At the same time organizations are asking three important questions:

1) Since I hear different things, what exactly *is* SOA, anyway?
2) If I use it, how is it going to help my organization eliminate cost, increase agility, deal with dwindling legacy skill sets, and address compliance?
3) Given that it has value, how do I get there?

This white paper addresses all three of these questions.

Many organizations are looking to SOA as a more cost-efficient and agile way to design applications.

What Is SOA?

SOA is a concept that has grown in meaning over the past few years. As a result, SOA has come to mean different things to different people.

SOA is an architecture for application design that has three main parts: Web service interfaces, service-based application components, and process orchestration.

Web services provide standards-based, flexible, and dynamic communication for SOA.

Web Service Interfaces

With the phenomenal demand for Web-based computing, Web services is a concept that was originally introduced to allow easy and flexible access to application components from Web environments. From the beginning, Web services have included a mechanism for communicating between applications. This mechanism became standardized via the Web Services Description Language (WSDL).

WSDL provides three basic concepts relative to application communication:

1) A standard syntax based on XML for invoking application components and passing data to them regardless of the language in which the underlying component is written.

2) Support for both synchronous (invoke and wait for reply) and asynchronous (invoke and don't wait for reply) communication.

3) Support for "loose coupling." When an application component is invoked via its WSDL name, a directory based on the Universal Description, Discovery, and Integration (UDDI) standard is examined to determine where the component really is and how to invoke it.

   SOA components are not "pre-linked" together—instead all invocation of components is dynamic with binding occurring at execution time. This means that as long as the calling "signature" remains the same, one component can be changed and replaced without touching any of the other components.

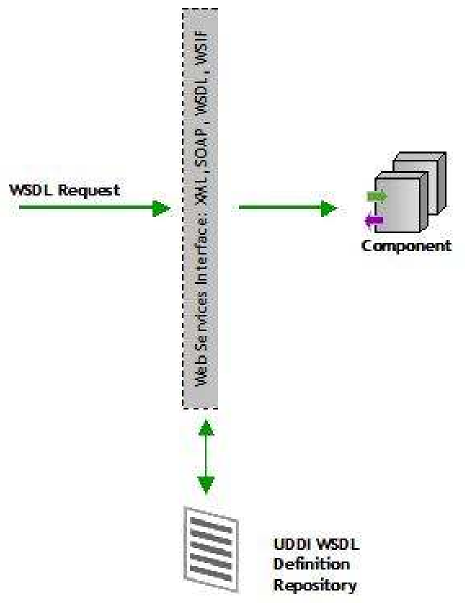
WSDL Request
Web Services Interface: XML, SOAP, WSDL, WSIF
Component
UDDI WSDL Definition Repository

Figure 1: Web services infrastructure

Although the infrastructure underlying Web services can be quite sophisticated in order to handle security, access policies, and remote access, the concept of a Web service as seen by an application designer is quite simple. What is most important about a Web service is that it is a standard form of communication accepted by all the major vendors—thus today virtually all packaged applications and technologies come with Web service interfaces.

Oracle provides numerous Web service interfaces for its applications and is looking to provide even more interfaces based on Web services in the future.

Although only a component of SOA, Web services all by themselves provide a number of benefits including the following:

1) They increase agility—the ability of IT to react to business change—by encouraging component-based development based on “loosely coupled” components that are easy to change and by encouraging the reuse of those components in future application development.

2) They can be used to “wrap” legacy application components without concern for the underlying programming language and environment in a way that makes the legacy components reusable—thus allowing the option of re-architecting the component later without changing any of the invoking applications. (Note: There is an inherent limit to this benefit—the legacy component must expose functionality in a reusable form, something legacy applications do not always do.)

SOA services are based on single business tasks.

Service-based Application Components

The second part of SOA is service-based components. In addition to Web services–based communications, SOA recognizes that the application components themselves need to be designed as services that perform individual business tasks such as

1) Adding an order to a database

2) Requesting credit card validation from a bank

3) Issuing a request to ship an order

SOA services represent a single business task in the application domain in question. Thus whatever an end user would see as a single “task” is a good candidate for a service. It is important not to confuse a business task with a complete business process. Business tasks are smaller and form single steps within a more-complex business process.

As a result of this, there are several “rules” to which service-based application components should adhere that increase their reusability:

1) Service-based application components should not remember things between invocations. This helps insure the components are single, complete tasks. (Note: For designers and developers used to object-oriented (OO) concepts, this is somewhat of a change since one advantage of OO objects is that object instances retain information from one method invocation to the next. In SOA, this concept can continue to be used within a service-based component, but not between components.)

2) Service-based components represent a single database transaction. Once a service-based component is invoked, it must “complete.” To undo what a service-based component has done, a “reversing” action will be required. For example, having booked a hotel via a “Book Hotel” service, undoing

the activity will require a "Cancel Hotel Booking" service rather than a rollback of the activity.

3) Since one of the goals of SOA is to create an architecture where "IT follows the business," service-based components are based on simple business tasks as defined by the end user rather than on technical concepts such as "functional decomposition."

Since one can argue that these are good general design principles even without SOA, it is worth noting that in some cases—although not all—existing legacy applications will have components that follow these rules well enough that they can be reused "as is" within an SOA environment. As we will see, if this is true then this fact can be leveraged when modernizing those legacy applications.

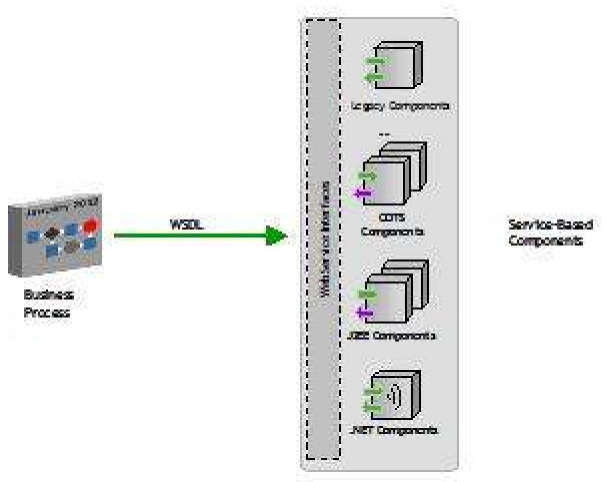

Figure 2: Service-based components come from many sources.

Application packages or commercial off-the-shelf (COTS) systems are also a source of service-based components. Since one of the goals of SOA is to eliminate the traditional "application silos" with reusable services, an application package that supports SOA must also support the same concept—it must be a set of services where the individual services can be used both within the package context and within custom applications. A COTS that utilizes this concept leads to a much higher degree of reuse and agility than a traditional set of APIs for accessing a COTS application silo.

The ability to orchestrate SOA services via process orchestration is a fundamental component of SOA that is often overlooked.

Process Orchestration

The third part of SOA is process orchestration. Given a set of services representing single business tasks implemented via service-based components and accessed as Web services, it is still necessary to "glue" the components together into complete business processes. This is accomplished via process orchestration.

Although in theory service-based components can invoke each other directly, a much more agile and lower cost environment is achieved if a process orchestration mechanism is used to script the various services into business processes.

A process orchestration engine that invokes SOA services provides a number of capabilities including the following:

1) The ability to detect various events, including end-user-initiated events, incoming external business-to-business (B2B) events, internal system events or time clock events, and then initiate the execution of a business process flow based on those events
2) The ability to invoke the service-based components that implement various business tasks that jointly implement a business process
3) The ability to invoke service-based components either synchronously (invoke wait for reply) or asynchronously (invoke and continue with process flow)
4) The ability to control the process flow based on various conditions
5) The ability to support parallel execution of tasks
6) The ability to support the execution of a process that will take hours, days, or weeks to complete, such as those that interact with other organizations

As with WSDL for Web services, there is an XML-based standard for expressing process orchestration in the form of Business Process Execution Language (BPEL). As with WSDL, all major vendors such as Oracle, IBM, and Microsoft support BPEL as way of orchestrating services into business processes within SOA.

Although not strictly required to support the orchestration of SOA services, the ability of the orchestration engine to support full workflow by also supporting interaction with end users is an important part of a complete SOA environment.

Rather than requiring users to manually track workflow and rely on the users to know when to turn to the computer to do work, extending orchestration to workflow provides the capability of having orchestration steps that place work in a user's work list, where it remains until the end user takes action. When the user selects the incoming item, the orchestration engine invokes the appropriate application—potentially interacting with the user—and when the user action is complete the orchestration engine continues with the next step in the process.

In this manner, a workflow work list is similar to using an e-mail inbox as a work list and in fact an e-mail client can act as a work list inbox for some workflow engines.

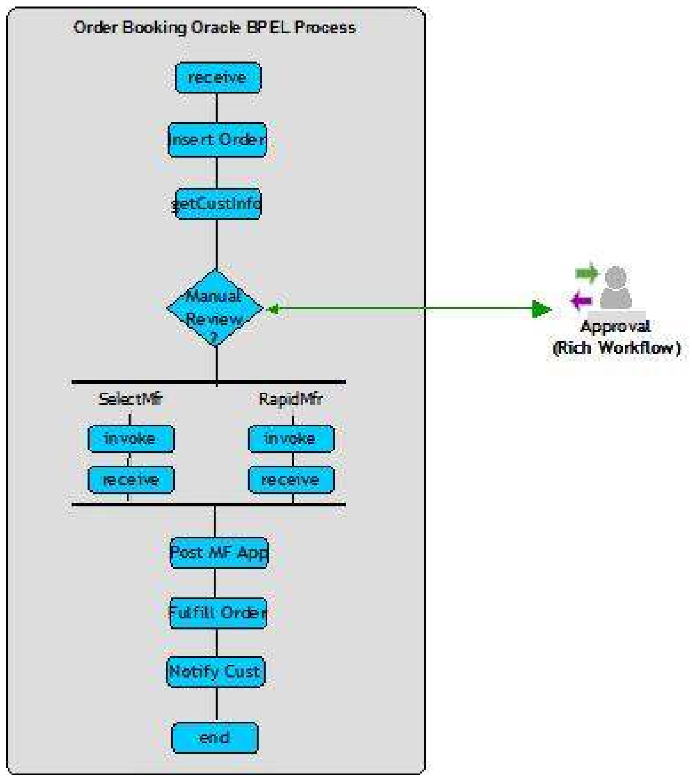

Figure 3: Process orchestration with user interaction added

SOA will yield substantial benefits for your organization.

How Will SOA Help My Organization?

There are many benefits to adopting SOA. Consider an organization that has transformed what was a legacy environment based on a mainframe to the complete SOA environment described above. Such an organization will have gained the following benefits:

1) Reduced operational cost.

   SOA service-based components can run on multiple hardware and operating system platforms including open source environments such as Linux. By making use of grid computing at both the database and application server level, SOA forms the ideal environment for virtual computing—an environment where applications see their surrounding environment as one, even though workload at all levels is begin automatically balanced among a number of low-cost and dynamically expandable platforms.

2) Reduced maintenance cost.

SOA encourages the use of services that reflect business tasks—thus lowering the cost of maintaining them. New technologies on more modern platforms make the development and maintenance of these services easier. Using process orchestration to isolate the process layer from the rest of the application means that the area of an application that changes most frequently—the process layer—is easier to change. The use of more modern technologies such as user-maintained business rules executed by business rules engines along with user-driven business intelligence systems further reduce the maintenance cost of SOA-based applications by allowing end users to maintain their own business rules and queries.

3) Increased agility.

By segmenting an application into orchestrated business-focused services, the technical implementation of business changes are easier to make. Changes that users perceive to be "easy" such as "from now on all purchase orders over $5,000 require a second level of approval" or "please send an e-mail to the head of shipping when there are more than 50 unshipped orders" no longer require weeks of testing since individual process flows can be changed without impacting all the services involved.

4) Increased compliance.

Adhering to compliance regulations is all about understanding and controlling your processes, who does them, and who changes them. With the clear separation of process orchestration, changes to processes and who can execute them can be automatically tracked and recorded for compliance audit purposes.

5) Low reliance on legacy skill sets.

SOA allows an organization to reuse existing legacy components—but it also provides the vehicle whereby individual legacy components can be transformed on an "as-needed basis" away from legacy technologies that are no longer adequately supported into more mainstream technologies. Over time such an architecture will allow an organization to make use of only modern, mainstream technologies for which staff can be more easily found.

SOA is a powerful architecture—but every organization has legacy applications preventing it from getting there.

How Do I Get There?

Having understood what SOA is and what the benefits of SOA are, it is worth noting that very few organizations are "SOA mature," in that they do everything based an SOA. One of the fundamental reasons for this is legacy applications—most organizations have existing applications running their business that were not structured based on SOA concepts.

These applications are often expensive to operate and maintain, not very agile, and reliant on legacy skills—but they are doing the job and no organization can afford to throw away their functionality. So a fundamental question arises: How do you get to SOA from where you are today? The answer is IT modernization.

What Is IT Modernization?

IT modernization is the process whereby legacy systems are transformed to a more modern SOA environment in ways that ensure that business content is not unintentionally lost. Technical content is maintained at various levels from a little to a lot depending on the type of modernization.

In contrast to green field development where specifications only come from users, IT modernization derives information from multiple sources such as current user workflow, current user enhancement requests and very importantly the current application itself. Different modernization approaches use each of these sources in various degrees.

IT modernization divides into two main aspects that interplay with each other: infrastructure modernization and application modernization.

Infrastructure modernization involves changing an underlying technology when there is either no change or very minimal change to the application itself. Examples of infrastructure modernization include migrating an application from UNIX to Linux or from UDB/UNIX to Oracle/UNIX. These types of migrations involve changing underlying infrastructure and may require such things as data migration, but the applications are by and large left unchanged.

Application modernization involves changing the application itself. Application modernizations span a wide range of complexity, from changing legacy file system calls within an application to Oracle SQL, to automatically transforming a legacy 4GL language such as Software AG's NATURAL to Java, to extracting business rules from legacy code for use in a business rules engine or for reengineering to new service-based components.

More information on why organizations look to modernize can be found in the Oracle IT Modernization Series white paper *Why Modernize*.

The Types of Modernization

An important thing for every organization to determine is which modernization approach to use—there are several with different costs and benefits.

As one can imagine, modernization in its full form is typically not a small task. As a result, it should also not be surprising that there are a number of modernization approaches that can be used—each with varying benefits and costs.

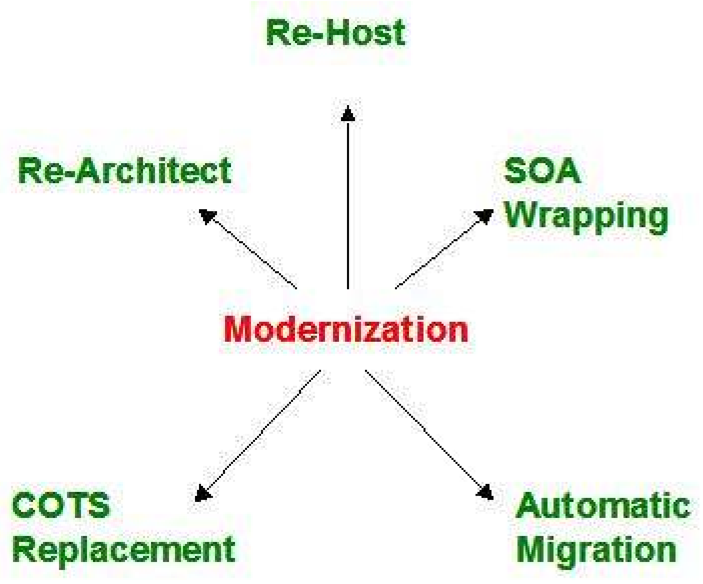

Figure 4: The types of modernization

The main types of modernization are:

1) Modernizing by re-hosting legacy applications.

   Shifting applications as is from environments such as legacy mainframes to Linux/UNIX in order to reduce legacy hardware and software costs. Re-hosting is primarily an infrastructure modernization, as re-hosting attempts to leave the application as is. However it can involve some application change if some aspects of the legacy environment are not supported in the new re-hosting environment.

2) Modernizing by wrapping legacy applications with SOA interfaces.

   Wrapping existing legacy components with Web service interfaces for use in an SOA environment. The legacy components may be on the original legacy platform or may have been re-hosted to a new platform. The quality of the resulting services will vary depending on the reusability of the current legacy components and on the degree to which the legacy components are truly service-based components.

3) Modernizing by automated migration of legacy applications.

   Automatically (90 percent or greater) transforming application code developed in legacy fourth-generation languages such as NATURAL to Java and transforming code embedded in legacy third-generation languages such as COBOL that access legacy databases and file systems to SQL accessing the Oracle database. Also included in this category is the automated migration of data that occurs when changing platforms or databases.

4) COTS replacement of applications.

   Replacing legacy applications with Linux/UNIX–based applications such as Oracle E-Business Suite, Oracle's PeopleSoft, Oracle's JD Edwards, and Oracle's Siebel.

5) Re-architecting applications.

   Transforming the current legacy application completely to maximize agility and minimize cost by

   a. Recovering core business logic from legacy applications and current user workflow into a technology-independent form

   b. Redesigning the application based on SOA design, process-driven design, and object-oriented design techniques as well as modern concepts such as business rules engines and business intelligence environments

   c. Reengineering the recovered information semi-automatically into service-based components based on platform-independent templates

   d. Regenerating the application from its platform-independent form to a technology-dependent form

More details on the types of modernization and the benefits and weaknesses of each can be found in the Oracle IT Modernization Series white paper *The Types of Modernization.*

An Oracle Modernization Insight is a key tool to help an organization answer its modernization questions and to develop its own road map to SOA through modernization.

The Oracle Modernization Insight: Determining Your Own Path to SOA

To begin the modernization process and to assist organizations in determining how best to use modernization to achieve a full SOA environment over time, Oracle offers a no-charge Oracle Modernization Insight.

The Oracle Modernization Insight brings in Oracle modernization experts to work with an organization to help determine that organization's best path to SOA through modernization. The goals of an Oracle Modernization Insight include

1) Understanding organizational business drivers, reviewing organizational decision-influencing factors, and gaining an understanding of the cost and risk of "doing nothing"

2) Examining the modernization options available in light of the specific organizational business drivers in order to lay out an organization-specific modernization road map to SOA.

3) Determining the desired target SOA that supports all modernization approaches consisting of a product-neutral architecture that can be implemented via a combination of Oracle and third-party supporting products—and where as a "hot-pluggable" framework other products can be substituted for Oracle products in cases where organizations have already invested heavily in other products.

4) Determining the need for further technical assessment. Depending on the modernization options chosen, some further detailed assessment of the current environment may be necessary. At the same time, it is important to avoid "analysis paralysis," where too much time is spent analyzing and not enough time is spent getting modernization results.

5) Determining any required partners. The Oracle Modernization Alliance (OMA) provides a source of potential modernization partners including system integrators that have modernization practices, specialty modernization vendors with powerful niche solutions, and vendors that add capability to implement the SOA Modernization Architecture. The Oracle Modernization Insight helps determine which partners—if any—are best for each specific situation. This also includes helping an organization determine who might lead the modernization project.

6) Working with an organization to develop the business case for the next step in your modernization process.

The Oracle Modernization Insight is a vital step in determining how to achieve SOA though Modernization. Since Oracle facilitates modernization but does not carry out modernization projects, Oracle's goal during the Modernization Insight is simple: helping to ensure that any modernization projects carried out minimize time, cost, and risk while still achieving the overall modernization objectives.

For more information on the Oracle Modernization Insight, contact your Oracle sales rep or contact Oracle's modernization experts at modernization_ww@oracle.com.

Walking through a real-life example helps clarify the costs and benefits of the various modernization approaches.

A Complete Modernization Example—Acme Widgets

In order to better understand why modernization is the best path to SOA, consider an example company called ACME Widgets. ACME Widgets sells widgets to customers directly as well as acting as a supplier to other companies that combine them with their own offerings for resale.

Oracle and ACME carry out an Oracle Modernization Insight. After a number of discussions and some onsite meetings, ACME has a much better understanding of its modernization options, whom it might partner with for parts of the modernization process, and what are the key business drivers.

As a result of the Modernization Insight, a number of factors come to light, including

1) ACME has been around for a long time and uses a number of legacy technology environments including a mainframe teleprocessing monitor (TP monitor) called CICS, the COBOL language, a mainframe DB2 database, the VSAM file system, and a fourth-generation environment sold to the company by Software AG that uses the NATURAL language and ADABAS database.

2) ACME also has many batch programs that run at night to synchronize systems and produce reports.

3) ACME's production data is quite complex, so ACME has also created a data warehouse maintained in an Oracle database on Linux for reporting purposes. The data warehouse is refreshed weekly.

4) The current ACME systems provide core capability very well. They handle special customer relationships and volume-buying discounts based on sophisticated algorithms that ACME has developed and which ACME does not want to lose. These algorithms have taken years to develop and are considered by ACME as valuable intellectual property, as they are part of what differentiates ACME from its competitors.

5) ACME is under pressure from both its end users and customers to make improvements in a number of areas including

    a) Providing the ability for customers to enter their own orders. Currently the customers either fill in forms that are sent to ACME or call ACME, who enters the orders and tracks customer information via an in-house developed customer relationship management (CRM) system. There is a transcription error rate of 5 percent in placing these orders that customers would like to see eliminated by allowing an option whereby they could place their own orders.

    b) Jones Trucking, one of ACME's major customers, has just implemented its own new system and wants to submit orders electronically.

    c) The ACME Finance Department would like to have real-time posting of all orders to accounting for better tracking. Currently orders are handled on the mainframe and only posted to the Oracle E-Business Suite accounting system running on Linux at night.

    d) ACME uses UPS for much of its shipping, and much time is spent entering the shipping orders. ACME wishes to make use of the UPSoffered SOA Web service interface so that shipment orders can be placed automatically.

e) ACME senior management would like better monitoring of the ordering process including knowing how many orders are in what status, which orders are waiting on back-order shipment, and how long key customer orders have been in process, so that ACME can escalate key issues more quickly with its suppliers.

f) ACME IT has a frozen IT budget, so the department would like to improve its service to the rest of the company—while reducing IT cost.

Having identified these factors and identified a path for ACME to achieve SOA during an Oracle Modernization Insight, let's walk with ACME down that path to see how the various modernization techniques address ACME's problems.

SOA wrapping is the least costly modernization technique and can be used to get some quick returns—although it can only go so far in attaining full SOA.

SOA Integration: Wrapping Legacy in Place with Web Services

As the ACME CIO is under pressure to deliver some initial capability very quickly, ACME has decided to begin its modernization journey by wrapping its existing applications with SOA interfaces—leaving them where they are on the mainframe but wrapping them so they can be used as part of SOA. This is a quick and low-risk place to start the modernization process based on integration.

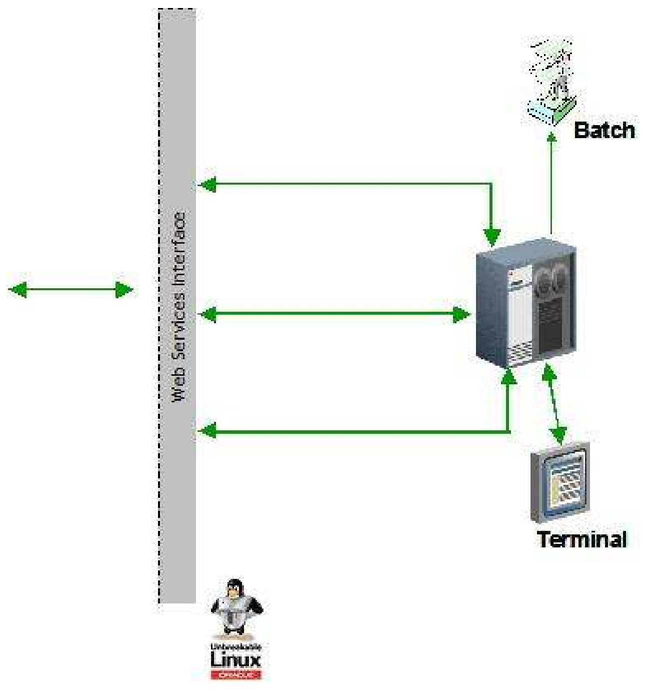

Figure 5: SOA Integration

The first modernization option is SOA wrapping that creates Web service interfaces to the legacy applications.

Creating the SOA Wrapper Web Service Interfaces for Legacy Components

ACME is targeting Linux as its long-term platform, so once it has wrapped its legacy as Web services, the company wants to orchestrate those services on the Linux platform in combination with other Linux J2EE application components. This means that it will require a means to communicate from the Linux platform to the mainframe and then invoke the legacy mainframe code as services.

After examining the requirements, ACME decides that the best approach is to use Oracle Adapters to wrap the legacy functionality as web services.

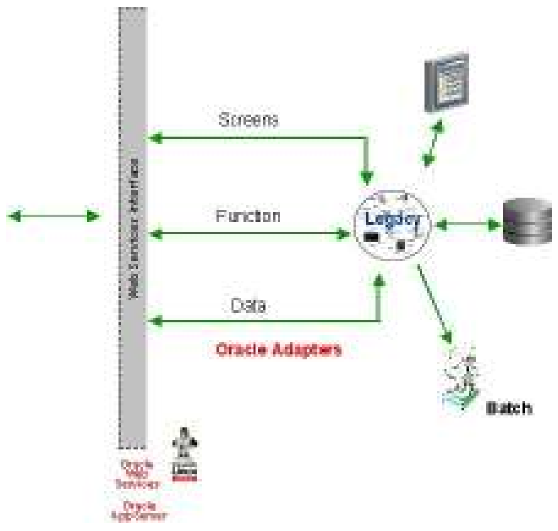

Figure 6: Oracle Adapters Provide Legacy Access

Having created Web service interfaces that map to existing legacy, ACME uses the Oracle Web Services Manager component of the Oracle SOA Suite to manage and secure access to the SOA-wrapped mainframe components. Oracle Web Services Manager stores the interfaces in an industry-standard UDDI format registry. Once these interfaces are stored in the registry, they are made visible and the wrapped components can then be invoked by other applications or can be combined into business processes using process orchestration.

The second step in SOA wrapping is to create business processes by orchestrating the wrapped services together with other new services.

Orchestrating the Services

Although SOA components can be invoked directly from any application, ACME recognizes that allowing various SOA components to invoke each other randomly will only lead to "loosely coupled" spaghetti systems rather than "tightly coupled" spaghetti systems. This will perhaps yield a slight improvement in maintenance cost as a result of the loose coupling, but it is not a good design strategy for maximum reusability and agility.

Therefore ACME decides to use process orchestration as a way to clearly separate process flow as a separate layer in its SOA.

In order to create business processes that make use of both the SOA-wrapped mainframe components as well as Java-based Linux components, ACME has decided to use the Oracle Business Process Automation (BPA) Suite to define its business processes. The Oracle BPA Suite generates industry-standard BPEL process definitions that are executed at runtime by the Oracle BPEL Process Manager that provides the capability to execute flows, interact with users, and manage the process flows—even if ACME executes over long periods of time.

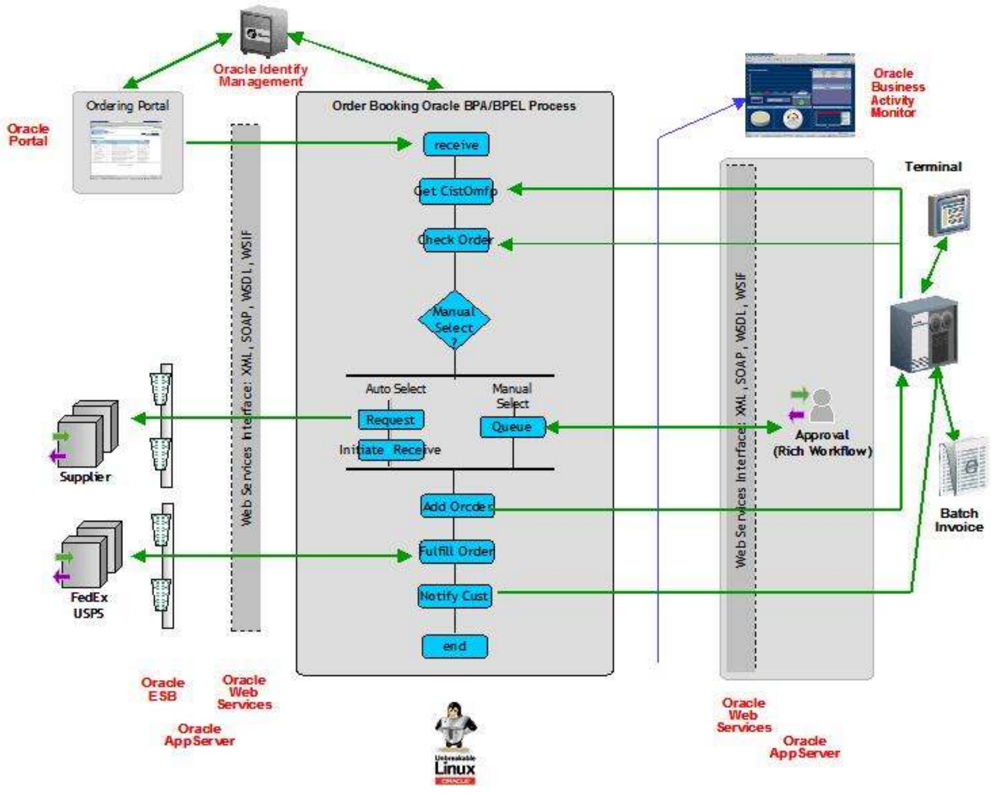

Figure 7: Orchestrating the SOA-enabled components

One major advantage for ACME of separating the process layer is that by allowing the legacy-wrapped components to be "hidden" from the users of the process flow it is now possible to quickly add a number of other modern capabilities to address a number of ACME's business problems, including

1) Using Oracle Enterprise 2.0 User Interaction & Portals to create Web-based portals so customers can have their own customized entry environments for entering their own orders. The portal interfaces can then initiate the newly defined process flows in the process flow layer.

2) Using the Oracle Enterprise Service Bus (ESB) component of the Oracle SOA Suite to provide an interface that allows Jones Trucking to send orders to ACME electronically. Both the portal interfaces and B2B interfaces invoke the same workflow, just from different "user interaction" viewpoints.

3) Using the Oracle Enterprise Service Bus (ESB) component of the Oracle SOA Suite to provide messaging and message transformation in order to

   a. Send asynchronous messages to suppliers and then create a new process flow to wait on an incoming reply via Oracle Enterprise Service Bus—a reply that may take days to occur

   b. Provide message transformation capabilities to transform internal ACME order message formats into the format required by a UPS-supplied SOA service that electronically places UPS shipping orders

4) Using the Oracle Business Activity Monitor (BAM) to monitor the various activities occurring within the process flows, including how many orders are at various stages of processing. The Oracle BAM dashboard allows this information to be displayed graphically, thereby allowing senior management to be alerted when order backlogs are becoming too large or when critical customer orders are being fulfilled too slowly.

5) Using Oracle Identity Management as a mechanism to provide a single point of authorization and authentication for the execution of process flows. ACME can combine Oracle Identity Management security with Oracle Web Services Manager security to insure that all process flows and service-implemented steps are properly secured. For wrapped services, this is additionally useful since mainframe security access can be replaced by a single-sign-on SOA-based security mechanism based on Oracle Identity Management and Web services management.

What ACME Achieves via SOA Wrapping

By just SOA wrapping existing legacy and then orchestrating the newly created services into processes, ACME has already achieved

1) The ability for ACME customers to enter their own orders

2) The ability for Jones Trucking to submit orders electronically

3) The ability to provide senior management with more up-to-the-minute information, improving business reaction time

4) The ability to place shipping orders to UPS electronically

5) Increased flexibility in ACME's own move to Linux by using orchestration to mix legacy components with Linux/Java components

The second modernization option is replacing legacy applications with commercial off-the-shelf (COTS) packages.

Replacing Legacy with Packages

Although ACME has achieved a number of benefits via SOA integration, cost savings is not one of them. Since the goal of SOA wrapping is to reuse legacy more or less as is, all current costs remain.

ACME is currently spending 80 percent of its mainframe budget on current systems operation and maintenance-—60 percent of this on operational cost and 40 percent on maintaining the legacy systems themselves. The ACME CIO is well aware that due to open source concepts and lower hardware costs, his per-unit Linux costs are much lower than the equivalent mainframe costs. Additionally, he does not want to carry the extra operational costs associated with having two environments—the mainframe and Linux. The ACME goal is that everything will be Linux-based in time.

To begin tackling this problem, ACME notes that it can replace the in-house-developed CRM components of its ordering system with a package. ACME chooses Oracle's Siebel CRM because it is a robust offering in the CRM market. ACME also chooses Siebel because of Oracle's plans for Oracle Fusion Applications.

Although Siebel CRM already provides current API interfaces in the form of SOA Web service interfaces, with Oracle Fusion Applications all Oracle applications are being fully redesigned into a larger, more-complete set of reusable SOA service-based components. This maximizes the reusability of purchased components by allowing the components to be used both as part of the packaged application as well as part of ACME's own applications. Oracle Fusion Applications components can then be scripted together with other SOA-based components using BPEL. This prevents Oracle Fusion Applications from being their own "application silos" that can only be reused in predetermined ways.

Additionally, Oracle Fusion Applications will deliver all their own metadata—including the process flows needed by the applications themselves—in the form of Oracle BPA Suite content. This means that the process flows and applications developed by ACME and those delivered as part of Oracle Fusion Applications can coexist not only in the same execution time environment but also in the same definitional environment.

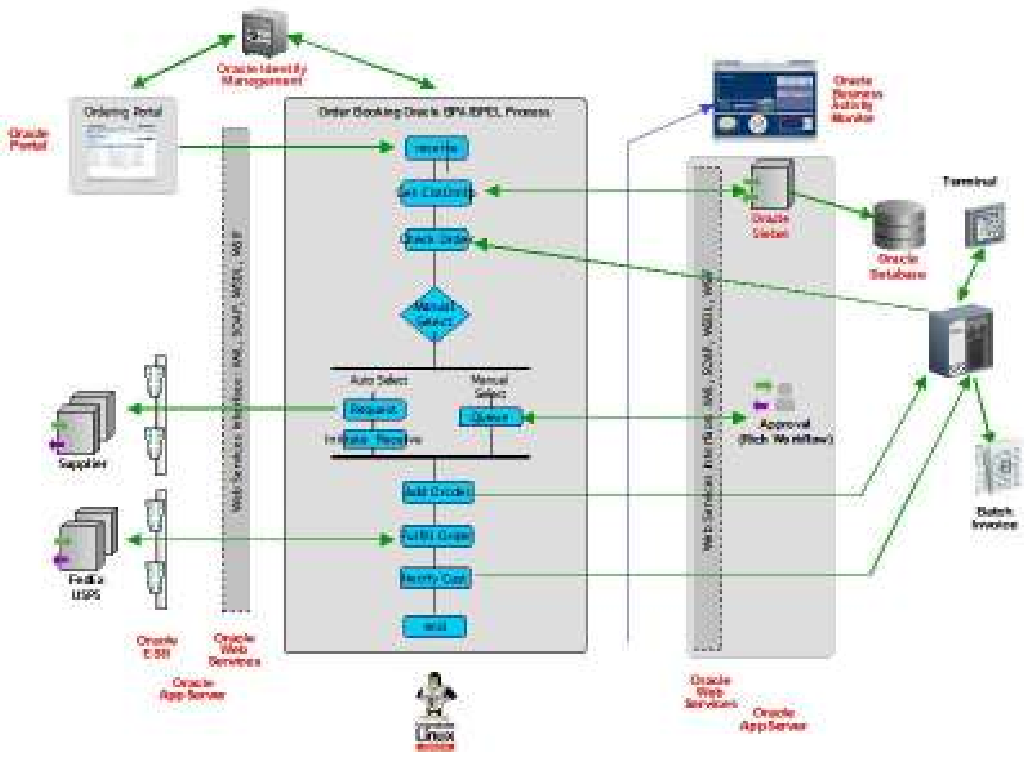

Figure 8: Replacing legacy CRM with Oracle's Siebel

What ACME Achieves via COTS Replacement

By replacing its legacy CRM code with Oracle's Siebel CRM, ACME achieves a number of benefits:

1) Lower operational cost by moving its CRM applications to a more cost-effective Linux platform.
2) Reduction of maintenance costs of the legacy CRM environment by using a commercial off-the-shelf application. Although Oracle's Siebel has maintenance costs, as a commercial package the costs are less than maintaining the legacy CRM environment.
3) Increased CRM functionality achieved by acquiring a "shared cost" package.

The third modernization option is to re-host applications on an as-is basis to a lower cost platform.

Re-hosting—COST Savings by Changing Platforms

ACME has achieved some cost savings by replacing its legacy COBOL CRM system with Oracle's Siebel, but the company still has a fair amount of COBOL applications left and it would like to get these applications over to Linux quickly to achieve further operational cost savings.

After examining its options, ACME decides to re-host its existing batch and COBOL/CICS applications onto the Linux platform. This will enable ACME to move its COBOL workload from the mainframe to Linux quickly without having to change the applications themselves.

Although ACME has a goal of eliminating reliance on legacy environments in general and CICS in particular, it also recognizes that by first moving the legacy to a lower-cost platform it can achieve a cost savings that will help the company fund a later, more-complex modernization effort to transform the CICS and COBOL to a J2EE environment.

In order to carry out the re-hosting project, ACME needs three things: a COBOL compiler for Linux, a CICS equivalent environment for Linux and a way to get from the legacy environment to the new environment. ACME chooses Micro Focus COBOL for the COBOL and Oracle Tuxedo as the replacement for the CICS environment. To carry out the migration, ACME examines the Oracle Modernization Alliance partner list and chooses MetaWare[1] since MetaWare can migrate the COBOL code to Tuxedo and can also eliminate VSAM calls replacing them with SQL. In this manner ACME achieves the elimination of a legacy file system even though the application code is left in COBOL.

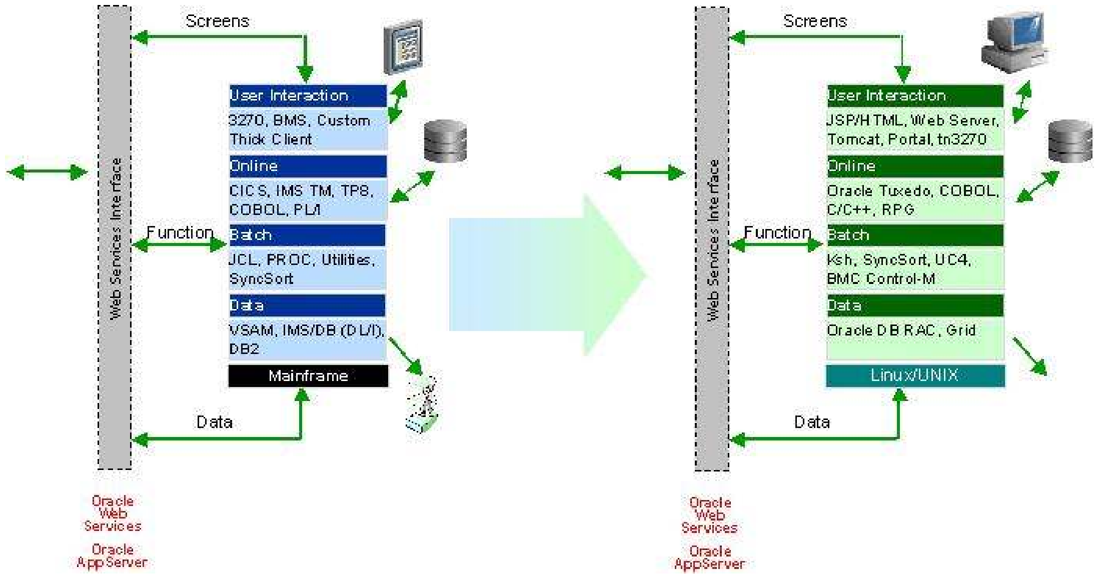

Figure 9: Re-hosting the remaining COBOL

[1] The partners referenced in this whitepaper are part of the Oracle Modernization Alliance within the Oracle PartnerNetwork Program; however Oracle does not endorse any of the partners or their software, solutions, services, or training noted in this white paper. Oracle disclaims any and all liability arising out of your use of the partners, software, solutions, services, and training mentioned in this white paper. All software, solutions, services, and training are provided "as is" and without warranty, unless provided by the authoring partner.

ACME also decides to eliminate the mainframe DB2 database by moving the DB2 data to an Oracle database as well, thus combining all production data in the same physical database as an Oracle data warehouse that ACME already has on Linux.

Since any access to the legacy applications done via ACME's previous SOA wrapping initiative uses Web service interfaces to access the application, any business processes defined using the Oracle BPA Suite and executing under Oracle BPEL Process Manager need not be changed as a result of the ACME re-hosting project. The only change required is to use Oracle Web Services Manager to note that the service is now running on a lower-cost Linux platform rather than the mainframe.

What ACME Achieves via Re-hosting

Having re-hosted its remaining COBOL applications to Linux, ACME has gained a number of other modernization benefits including

1) Lowering operational costs by executing on a lower-cost Linux platform and eliminating the cost of mainframe software
2) Eliminating the need for remote communications when accessing the legacy COBOL components, as they now run on the Linux platform as well
3) Combining all of ACME's data into a single virtual Oracle Database that can be implemented via an
4) Providing a single source of data that can be used to more easily produce an Oracle data warehouse
5) Provide a single database for both production and data warehouse data, thus maximizing the flexibility in establishing end-user access via Oracle Business Intelligence

The fourth modernization option is to automatically transform application code—useful for making database changes and eliminating fourth-generation languages.

Eliminating NATURAL/ADABAS with Automated Migration

ACME has now further reduced operational costs by moving its COBOL applications to a Linux platform, but due to the NATURAL applications using the ADABAS database it still needs its mainframe platform. The NATURAL programs implement a number of things, including printing the invoices that are sent to customers.

Since the lease on its mainframe is coming up in six months, ACME decides it would be highly beneficial to convert the NATURAL/ADABAS environment to a Java/Oracle environment as quickly as possible to avoid renewing its mainframe lease, so the company chooses an automated migration approach.

ACME realizes that the maintenance costs of the resulting application will not change very much since an automated migration process is likely to increase the number of lines of code slightly and the resulting Java code will be object-oriented code, but not an object-oriented design.

Since automated migration achieves its automation by doing largely a one-to-one line-by-line transformation, the design of the resulting application will still largely reflect the original NATURAL design.

Although this is recognized as not being a perfect solution in achieving object-oriented SOA service-based components, ACME notes that since the resulting Java code looks "NATURAL-like" this will actually increase the likelihood that the company can retrain its existing NATURAL developers to work with the new Java application. Also, the automated approach will mean legacy 3270 terminal interfaces can be transformed into graphical interfaces that will still function largely as they did before—therefore not requiring end-user retraining.

In the end, ACME decides to opt for the immediate cost savings that can be gained by quickly leaving the mainframe completely and leaves any further re-architecting of the resulting Java code into better service-based components for a future modernization step.

Fourth-generation languages and databases such as NATURAL/ADABAS, IDEAL/Datacom, and ADSO/IDMS are good candidates for automated migration.

Eliminating NATURAL/ADABAS

In order to carry out the transformation, ACME again looks to the Oracle Modernization Alliance and chooses BluePhoenix Solutions[2] DBMSMigrator for ADABAS/Natural to carry out the transformation. Using DBMSMigrator for ADABAS/Natural, BluePhoenix develops a data model mapping from the non-relational ADABAS database to the relational Oracle Database. DBMSMigrator for ADABAS/Natural is then used to transform the NATURAL code to Java in a highly automated manner, changing the code to reflect the new data model at the same time.

[2] The partners referenced in this whitepaper are part of the Oracle Modernization Alliance within the Oracle PartnerNetwork Program; however Oracle does not endorse any of the partners or their software, solutions, services, or training noted in this white paper. Oracle disclaims any and all liability arising out of your use of the partners, software, solutions, services, and training mentioned in this white paper. All software, solutions, services, and training are provided "as is" and without warranty, unless provided by the authoring partner.

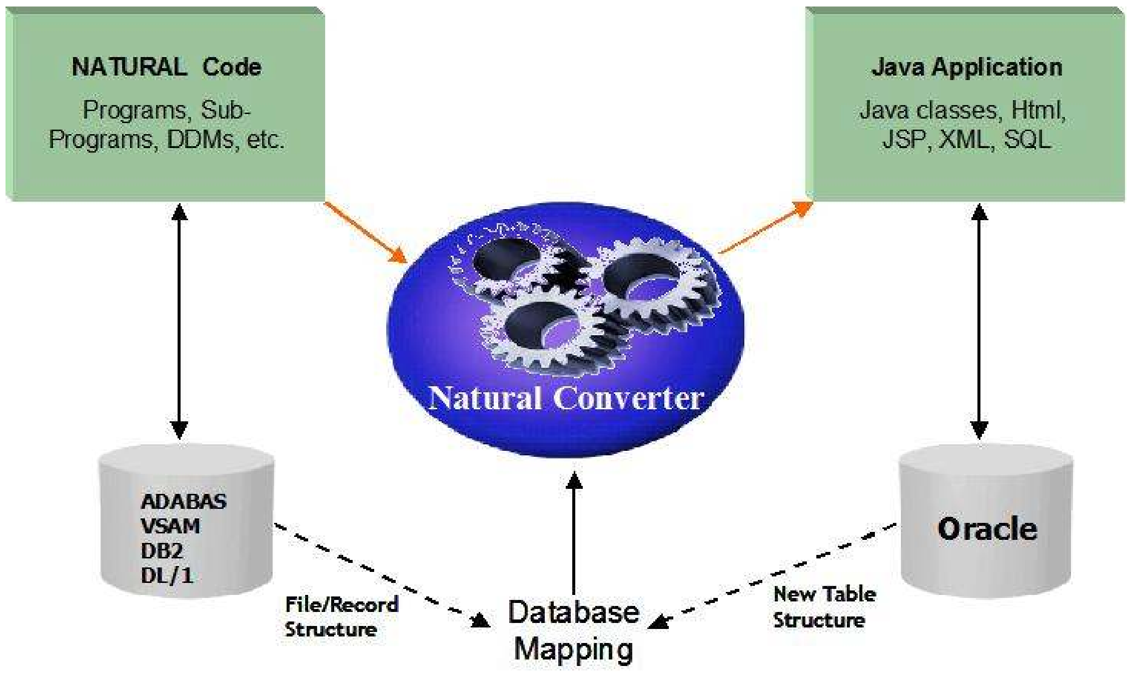

Figure 10: BluePhoenix automation of NATURAL/ADABAS to Java/Oracle

Data migration programs used to migrate the ADBABAS data to Oracle Database are also generated from the data model mapping. Since the transformation process is highly automated, it can easily be tuned and then rerun repeatedly—providing for a short "final transformation" downtime window allows the inclusion of any application maintenance changes made to the existing NATURAL code. ACME also uses BluePhoenix Solutions to transform any batch JCL used to run NATURAL programs to Linux script.

Having completed the automated migration of NATURAL/ADABAS to Java/Oracle, ACME has now eliminated the need for the mainframe while also continuing on its path of achieving a full SOA environment on Linux.

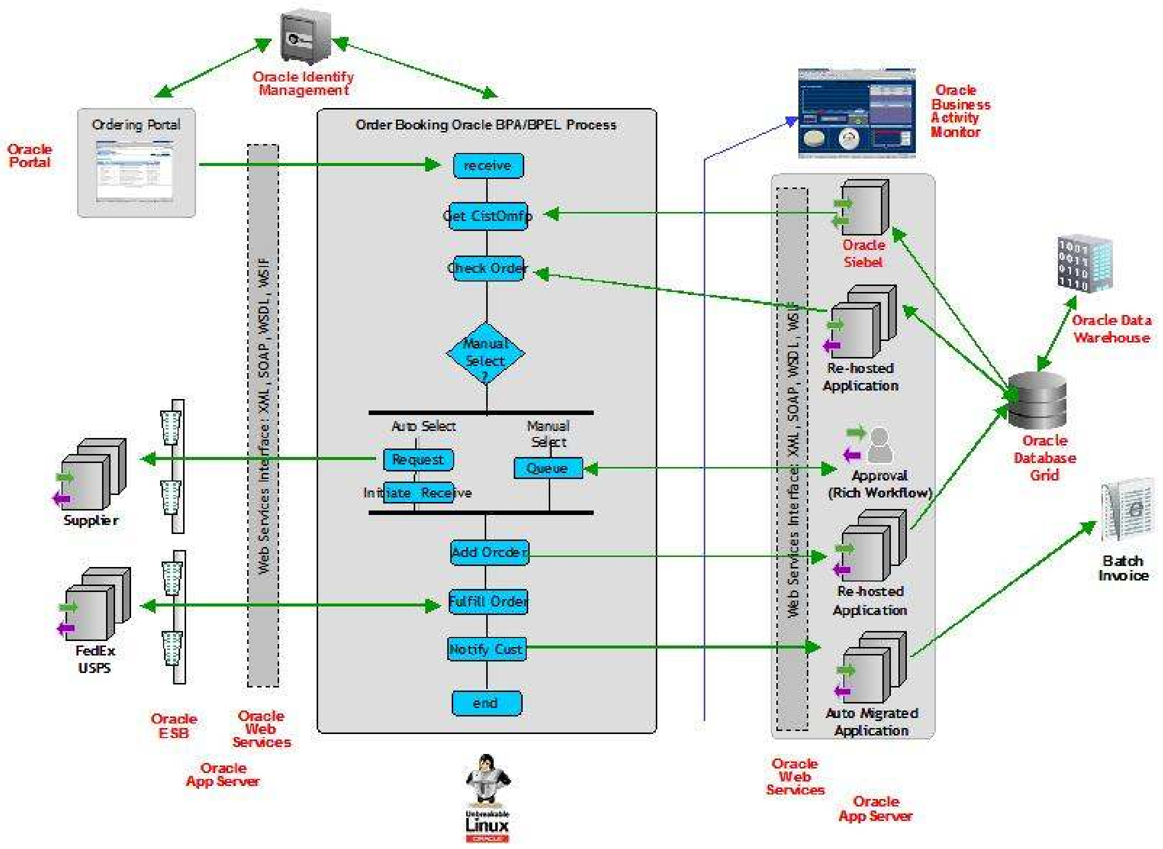

Figure 12: ACME ordering after automated migration

What ACME achieves via automated migration

After carrying out the automated migration of NATURAL/ADABAS to Java/Oracle le, ACME has achieved another set of benefits, including

1) Further operational cost reduction by eliminating the mainframe hardware.
2) Further operational cost reduction and reduction in reliance on legacy skill sets by eliminating the need for mainframe skill sets.
3) Eliminating the cost of the legacy ADABAS/NATURAL environment by replacing with more cost effective Oracle Database and Java.
4) Eliminating reliance on legacy ADABAS/NATURAL skills.
5) All data is now stored in a single Oracle Database Grid making online queries and business intelligence access to data much simpler.
6) Since all production data is now stored in an Oracle database, ACME can now integrate all production data to its data warehouse without having to

worry about the cost and effort of converting the data from legacy file systems and databases.

The fifth modernization option is to re-architect to maximize agility and minimize cost.

Re-architecting the COBOL Applications

ACME has managed to eliminate its mainframe, and as a result its operational costs have dropped—however application maintenance costs remain high. The simple reason for this is that the re-hosted COBOL applications are still the same as they were and the automatically migrated NATUAL—although now in Java—is actually slightly larger than before in terms of lines of code.

ACME has also noticed that although it has improved agility by adding a process layer early on in its modernization efforts, the services exposed by the COBOL and automatically migrated NATURAL do not match the services that ACME would have desired had it "started over." The COBOL and migrated NATURAL applications are too monolithic, and more flexibility and reuse can occur if the applications are broken down into smaller, service-based components focused on business tasks by eliminating any legacy technical structure so the ACME can maximize its IT agility in reacting to business change.

ACME also notes that some of the functionality accomplished by the current applications could be accomplished much more effectively by using modern concepts such as business rules engines and business intelligence systems, where end users have direct access to change their own business rules and reports. This would result in a decrease in the number of lines of custom code and hence a decrease in maintenance costs.

Re-architecting consists of four main steps: recovering content, creating a new design, reengineering the recovered content and generating new code that can be functionally the same—but architecturally very different.

So ACME decides to re-architect the re-hosted COBOL as well as the Java code transformed from NATURAL following the four main re-architecting steps:

1) Recover core business content from end users as well as the current applications
2) Create a new SOA-based design that can incorporate potential functional changes as well as new concepts such as business intelligence systems, report writers, and business rules engines that replace legacy code
3) Re-factor the recovered content to the new design using model-driven modernization (MDM) tools and methodology that map recovered legacy content to either new "replacement" technology or modernized use cases to drive new business processes
4) Regenerate the application using templates and generation techniques in a manner so as to reduce future maintenance.

In order to carry out the re-architecting process, ACME once again turns to the Oracle Modernization Alliance and selects MAKE Technologies to use its Transformational Legacy Modernization (TLM) approach for re-architecting applications. TLM combines a re-architecting methodology with a set of tools to implement the methodology in order to implement the recover, redesign, re-factor, and regenerate phases of the re-architecting approach.

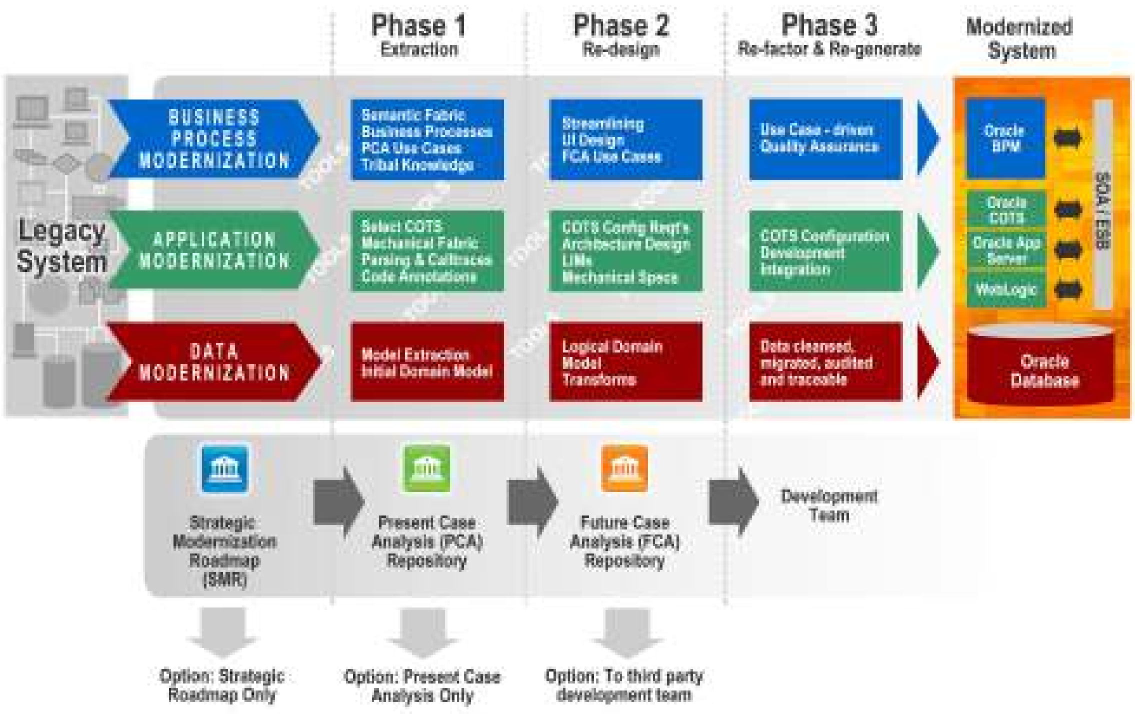
Phase 1
Extraction
Phase 2
Re-design
Phase 3
Re-factor & Re-generate
Modernized System
Legacy System
BUSINESS PROCESS MODERNIZATION
APPLICATION MODERNIZATION
DATA MODERNIZATION
Semantic Fabric Business Processes PCA Use Cases Tribal Knowledge
Streamlining UI Design FCA Use Cases
Use Case - driven Quality Assurance
Select COTS Mechanical Fabric Parsing & Calltrace Code Annotations
COTS Config Req'ts Architecture Design LIMs Mechanical Specs
COTS Configuration Development Integration
Model Extraction Initial Domain Model
Logical Domain Model Transforms
Data cleansed, migrated, audited and traceable
TOOLS
Oracle BPM
Oracle COTS
Oracle App Server
WebLogic
SOA / ESB
Oracle Database
Strategic Modernization Roadmap (SMR)
Present Case Analysis (PCA) Repository
Future Case Analysis (FCA) Repository
Development Team
Option: Strategic Roadmap Only
Option: Present Case Analysis Only
Option: To third party development team

Figure 12: MAKE Re-architecting approach

Although re-architecting makes use of technology to assist in the re-architecting approach, in order to accomplish the complete change to a fully process-driven and object-oriented design based on SOA, human intelligence is required, and as such the re-architecting process in general and the re-factoring phase in particular will be semi automated at best.

This naturally increases the risk over some of the other modernization techniques—but at the same time a completely new application design can be achieved.

As part of the re-architecting process, ACME is able to make a number of further changes to its applications, including

1) Removing the order discount algorithms from the legacy code and implementing them in the Oracle Business Rules Engine, where end users can define and manage the rules themselves

2) Eliminating the legacy batch processes that were using an “enter the data in the day, process at night” approach with online processes

3) Eliminating batch processes such as printing and mailing invoices with e-mailing of invoices directly

4) Re-architecting the wrapped monolithic legacy programs to break them into smaller, more business-task-oriented service-based components

5) Using Oracle Business Intelligence Suite to replace many of the batch reports with online reporting and analysis

6) Extending their process layer by removing workflow such as error notification still embedded in the legacy code and incorporating this work flow into the workflow layer maintained in the Oracle BPA Suite.

7) Transforming all application design as well as implementation to an object-oriented and process-driven design approach

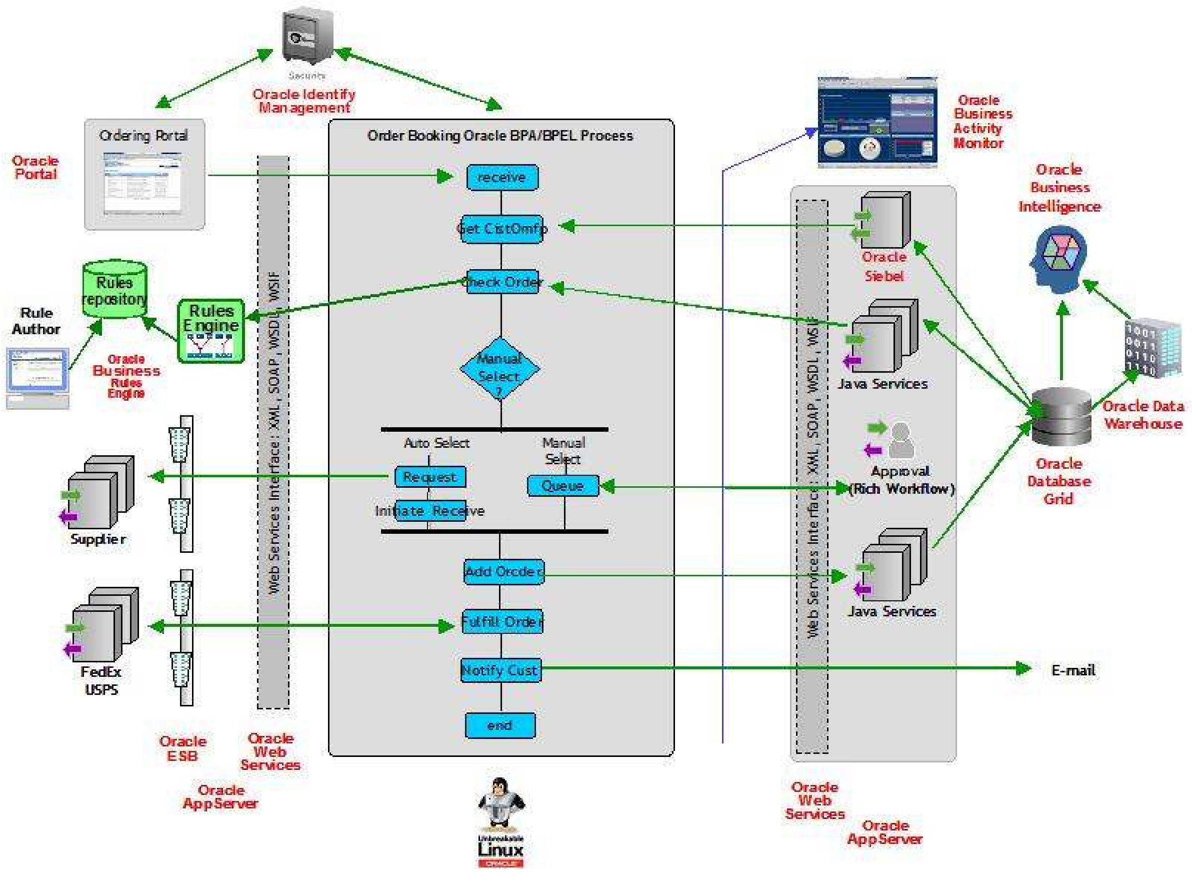

Figure 13: ACME ordering following re-architecting

What ACME Achieves via Re-architecting

After carrying out the re-architecting process, ACME has achieved another set of benefits including

1) Maximizing business agility by creating service-oriented components that reflect business tasks.
2) Minimizing maintenance cost by making use of concepts such as business intelligence and business rules to put application change in the hands of the end users.
3) Reducing application maintenance costs by decreasing the number of lines of custom code in the applications,
4) Eliminating reliance on legacy skill sets by transforming completely to Java/J2EE.
5) Becoming more compliance-oriented by totally separating the processes that drive compliance from the rest of the application.

Determining Your Specific Modernization Path

Each organization will have its own modernization road map to SOA and the Oracle Modernization Insight is the best approach to help an organization determine that road map.

The ACME Widgets example is somewhat contrived in order to demonstrate all the modernization options in one example. Despite this it is actually a perfectly reasonable modernization road map to SOA for ACME and could be followed.

Still, it is important to realize that each organization will have to determine its own modernization road map to SOA based on its own business drivers, legacy technologies, aversion to risk, and other business factors.

For example, in the case of ACME some other options might have been chosen, including

1) Not bothering with SOA wrapping and moving straight into the techniques such as re-hosting and automated migration that get load away from the mainframe quickly. This might be desirable if there is a time constraint on the lease of the mainframe platform
2) SOA wrapping the existing legacy on the mainframe, and then not bothering with re-hosting and instead moving directly to re-architecting the legacy directly from the mainframe to a full SOA environment with service-oriented components
3) Re-hosting the COBOL but not bothering with automated migration of the NATURAL/ADABAS environment and instead re-architecting it directly to a full SOA environment with service-based components
4) Replacing the in-house legacy CRM system with Oracle's Siebel as a first step rather than a second step

5) Replacing the in-house legacy CRM with Siebel and re-architecting the rest of the legacy in parallel. The fastest and least expensive to get to a complete SOA and its associated cost savings—and the most risky

Each of the above approaches is feasible. Which one is best depends on an organization's specific business drivers and legacy technologies—and the Oracle Modernization Insight can help an organization determine the best path.

Doing nothing may seem like an option—but it really is not.

The Risk Of Doing Nothing

Unlike issues such as Y2K that have hard deadlines, organizations may make the mistake of thinking that there does not appear to be the same compelling need for modernization. Some organizations consider modernization to SOA as "very complex" and "very risky" and as a result might consider doing nothing.

In reality, no organization can afford to ignore IT modernization for a number of reasons:

1) As seen with the ACME example, modernization to SOA is not just about the high risk of "rip and replace" or "rewriting." Some modernization techniques are low-risk to implement and an organization can create a phased approach to modernization. And all the modernization techniques reduce risk by using the current applications as starting points.

2) Organizations must become more agile in order to keep up with business competition. An organization that does not consider modernization while a competitor does will find itself at a disadvantage in the market.

3) There is a large downside cost risk if an organization does nothing. Currently estimates are that up to 80 percent of IT budgets are spent on operations and maintenance of legacy applications—and this number is projected to grow to 85 percent in the next few short years. Doing nothing means facing the risk of either increasing IT budgets or allocating an ever-decreasing proportion of budget to ongoing business needs.

4) Organizations with shared-cost legacy environments that do not modernize will find their proportion of the cost steadily increasing as other organizations that share the cost of the environment do modernize. Being the last one standing in a shared-cost environment is a very expensive proposition.

5) There is a large downside skill risk if an organization does nothing. Today's younger generation of IT staff are not trained in legacy environments and despite the efforts of some vendors to claim more will be trained—the bottom line is that when it comes to legacy skills, retirement will dramatically exceed replacement in the years to come. This is already becoming a major problem with less-used legacy technologies such as fourth-generation languages and will, in time, also be true of more-mainstream legacy technologies as well.

An organization need only check with the HR department regarding retirement of its current personnel to determine when this will become an issue.

6) Compliance will become a greater issue in time. Today's compliance regulations are only the beginning of future regulatory demands. One of the issues of compliance is that it creates extra cost for organizations—with no associated revenue. Organizations have to minimize the cost of implementing compliance and monolithic legacy environments are not designed to incorporate the adaptations needed for compliance.

IT modernization involves an investment to achieve future cost savings. But the alternative to investing today when there is still time to define an optimal modernization road map to SOA is to pay twice as much tomorrow.

There are different approaches to achieving SOA through modernization and a full SOA environment can be accomplished over time. But unless organizations are willing to risk the same "last-minute crunch" that occurred with Y2K, they need to start examining their options today.

Conclusion

Modernization is a key mechanism for achieving SOA, and every organization should start today in developing its modernization road map to SOA.

IT organizations are under increasing pressure to reduce costs while at the same time increase their ability to react to ongoing business demands. To achieve this they need to take advantage of the cost savings and flexibility of a service-oriented architecture (SOA).

In order to achieve SOA, organizations will have to turn to IT modernization as the path to SOA—a path that allows a phased transformation of legacy applications to SOA while still reusing and recovering their business content.

Every organization needs to consider SOA and since no organization can afford to discard the intellectual property contained in its existing applications, IT modernization is the ideal path to SOA—and Oracle is here to help.

The Oracle IT Modernization Series: *Modernization—The Path to SOA*
August 2008
Author: Ted Venema
Contact: modernization_ww@oracle.com
For more information: www.oracle.com/goto/modernization
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com