

Application Upgrades and Service Oriented Architecture

*An Oracle White Paper
April 2008*

Application Upgrades and Service Oriented Architecture

EXECUTIVE SUMMARY

Maintenance costs, for major point releases as well as more frequent patches, is a topic that catches the attention of most ERP and CRM customers. In this whitepaper, we will provide a detailed look into why and how SOA-enablement will reduce cost and risk of Application upgrades. This will help you accurately understand the lifetime value of investing in SOA technology now over a “like to like” upgrade. You will see why Oracle’s recommendation to get on the “Path to Fusion” (www.oracle.com/applications/evolutionary-path-to-fusion.pdf) will save you maintenance dollars today.

INTRODUCTION

In a recent Forrester study, upgrades made the top concerns list of ERP customers together with Improving Integrations and Shifting the Process Orientation. This is not surprising, given that Applications upgrades are prone to cost overruns, delays and unexpected outages. Across a significant sample size, we found that upgrade costs for major point releases averaged 18-20% of an application’s initial implementation cost. With a typical point release brought to market every 2 years, that’s a significant budget item and clearly a reason to think about ways to reduce this cost.

So, why is the promise of COTS (“commercial off the shelf”) applications – reduced time to market and IT costs via ready-made functionality – tainted by upgrade headaches? What’s the culprit that’s affecting the cost of application maintenance? Most importantly, what can you do to avoid high lifecycle costs for your application deployment?

Carnegie Mellon’s Software Engineering Institute addresses this topic head-on in its paper “Evolutionary Process for Integrating COTS-Based Systems (EPIC)”. It finds that a narrow focus on functionality combined together with shortcuts taken to customize and extend off-the-shelf application functionality is at the core of the problem. As an aide to remedy the situation, it believes that application users need to look beyond pure functionality and include custom code, linkage/interfaces to third party systems as well as business processes into their implementation and

“Application upgrade costs for major point releases average 18-20% of an application’s initial implementation cost”

— Oracle Consulting

“Many organizations find that COTS-based systems are difficult and costly to build, field and support.”

— Evolutionary Process for Integrating COTS-Based Systems (EPIC)
<http://www.sei.cmu.edu/cbs/epic>

lifecycle methodologies. Fundamentally, Carnegie Mellon as well as other experts believe that proper Enterprise Architecture was often neglected during implementations of packaged applications in the past.

“The real win with Service Oriented Integration, therefore, is in the dramatic reduction of cost at the maintenance and change phases of integration”

— ZapThink: Understanding the Real Cost of Integration

But there is hope on the horizon. Service Oriented Architecture (SOA) with its approach of “loosely coupling” has demonstrated that it can reduce maintenance costs. For example, a study by SOA Industry Analyst Zapthink shows that maintenance and change costs of SOA-based integrations, architected properly, are dramatically lower than traditional integration approaches. The additional good news is that core ERP and CRM Applications are now “service enabled”, meaning they feature standard web services that can participate in a SOA. This includes PeopleSoft (starting with PeopleTools 8.46), Siebel (starting with Siebel Tools 7.8), JDE Enterprise One (starting with Tools 8.97) and E-Business Suite 11i (with the Oracle E-Business Suite Applications Adapter).

THE BUSINESS CASE FOR SERVICE ENABLEMENT

This section will help you understand the impact that SOA enablement has on the economics of an application upgrade. We will demonstrate how SOA enablement can reduce the significant task of validating existing interfaces and customizations in the later phase of an application upgrade.

In general, the upgrade process for a major point release, which in our experience costs 18-20% of the initial implementation cost, can be broken into two phases:

- The process of applying upgrade scripts and testing core application functionality followed by
- Testing and potentially re-working customer-specific customizations and integration interfaces. Re-work includes re-establishing proper behavior as well as also re-tuning for performance.

Clearly, phase two will vary based on the number of customizations and integrations with external systems. In practice, we have found many cases where testing became the most expensive part of the entire upgrade. A 50% split between the two phases is typical. Very rarely have we seen the testing phase contribute less than 30% of the overall costs.

When thinking about your individual application deployment, keep in mind that, according to Carnegie Mellon’s Software Engineering Institute, the time needed to conduct test and integration activities is often grossly underestimated. For example, one upgrade scoped testing efforts at 1,600 hours and easily spent 10,000 hours.

To what degree can SOA-enablement reduced the cost of the phase two for testing and potential re-work? One customer, a Fortune 100 company, explained that before SOA, unexpected issues were often discovered late during the upgrade cycle.

Data Points:

- **Upgrade costs: 18-20% of initial implementation costs**
- **50% of the upgrade process related to validating customizations & integrations**
- **SOA reduces testing & re-work by 50-75%**
- **Total upgrade cost reduction with SOA: 15-38%**

“The time needed to conduct test and integration activities is often grossly underestimated. For example, one upgrade scoped testing efforts at 1,600 hours and easily spent 10,000 hours”

— SEI, Carnegie Mellon

Due to interdependencies of complex integration code, attempts at fixing these issues created new issues, causing ripple effects. After SOA enablement, the customer reported testing and correction cycles reduced from 3 months to 3 weeks with SOA, a 75% time reduction. Simply spoken, SOA-based integration helped this customer scale down the effort of a large-scale integration test to about the effort associated with a unit test. We will later explain the specifics of this “order of magnitude” impact in detail, but the customer attributed most of the benefits to:

- Extensions layered on top of well-defined service interfaces that are easier to test
- Dependencies removed through loosely coupling which accelerates root cause analysis
- Service orchestrations implemented declaratively, not programmatically, which facilitates the debugging process

Please keep in mind that the impact of SOA is more profound on validation and re-work than on the initial implementation effort of integrations. That’s a fundamental insight shared by experts such as SOA industry analyst Zapthink. SOA-based integrations might require some additional architecture overhead initially, but will lead in the long run to dramatically lowered maintenance and change costs. Hence, we can treat experience from customers such as Helio, who were able to reduce the development time of their initial integrations with a SOA-based approach, as a lower bound. Compared to implementing integrations between E-Business Suite and an outsourced logistics provider point to point with PLSQL – an estimated 8 months effort. Helio was able to reduce its integration timeline by 50% to 4 months using Oracle SOA Suite.

These data points suggest that you should be able to reduce the testing phase of your application upgrade by over 50% if you make use of SOA principles for your customizations, extensions and integrations. The total cost impact on the entire upgrade will therefore be as high as 38% and at a minimum 15%. Given that you face such a point release upgrade every 2 years at a cost of 18-20% of you initial application implementation cost, you see that a SOA-enablement strategy makes for a very sound long-term investment.

Furthermore, you should be aware of two additional SOA-enablement benefits, which are more difficult to quantify, but important for practical upgrade cycles:

Further SOA benefits:

- **Reduced dependency on third party application staff**
- **Reduced risk of application downtime**
- Reduced dependency on third party application staff - One customer regarded as a key benefit that the upgrade testing cycle could happen largely independent from Database Administrator (DBA) staff serving the external partner applications. Before, traditional, more intrusive, integrations required the external system DBAs to be closely involved. In terms of cost, one application’s upgrade also affected the time of the

DBAs associated with external systems. SOA-enablement reduces these dependencies.

- Reduced risk of application downtime - The business case of one large Telco customer to SOA-enable its integration interfaces was entirely built on reducing downtime risk associated with test and correction cycles when applying bi-yearly patches. In certain cases, customers even defer patches to avoid such downtime risk. In the worst case, this shortcut can defer important regular application maintenance to the point that the only option is a very costly re-implementation because of having deviated too significantly from the recommended upgrade path. SOA-enablement can avoid such a scenario by reducing the risk of smaller patches.

WHAT MAKES APPLICATION UPGRADES COSTLY?

By now, we know that experts believe a narrow focus on functionality combined together with shortcuts taken to customize and extend off-the-shelf application functionality is at the core of costly upgrades. Carnegie Mellon's Software Engineering Institute urges applications users to prioritize the following Enterprise Architecture topics in order to minimize lifecycle costs:

- Custom code required to adapt the off-the-shelf functionality
- Linkage/interfaces to third party systems
- End-user business processes that span multiple applications

Let's first understand how an application upgrade process can be negatively impacted when shortcuts are taken for such Enterprise Architecture topics:

Customizations, Bolt-On Applications

As off-the-shelf functionality of packaged applications never matches requirements 100%, many application deployments are significantly customized or extended using native application technology or bolt-on extensions. For the same reasons, best-of-breed niche applications are purchased to complement missing domain-specific functionality.

Unfortunately, such customizations and extensions can interfere with the evolutionary upgrade cycle of the packaged application to meet new market requirements:

- For customizations, the key issue is tight coupling to the application's database schema. If the design of the customization is intrusive, meaning it directly accesses the application's database, changes that a new release introduces to the database schema can cause the customization to fail.

"57% of companies not running the current release of their ERP indicate the prohibitive cost of upgrading customizations as a key reason"

— Aberdeen Group: Best Practices in Extending ERP

- Complementary niche applications, introducing their own databases, require data replication. This creates redundant data as well as additional integration points that can cause trouble if implemented as shortcuts (see below).

We have observed application implementations with hundreds of customizations and multiple complementary niche applications. The results are long validation and re-work cycles in the later stages of an upgrade.

Brittle Integrations and Data Replications

As priority is on application functionality, integration design unfortunately becomes an afterthought. Accordingly, Aberdeen Group comments in its “Achieving more value from Enterprise Applications” benchmark report: “in many enterprises, business processes consist of silos of application software connected by the slap-dash integration equivalent of duct tape, chewing gum, and string”.

Ultimately, any upgrade is not successfully completed if only a single integration interface fails. This is a tough goal given that such integrations are often created in ways that make it very likely for them to fail:

- Integrations are completed as over night batches, an approach that is less resilient when it comes to exception and error handling.
- Integrations are implemented via database synchronization (e.g. DBLink) at the lowest level of the technology stack. While performant, such integration design is not upgrade-resilient at all as the database schema might change during a major point release. Also, available built-in error handling in the application business logic is circumvented.
- Integrations are implemented “point to point”, replicating data between two application databases. In the setting of multiple interconnected applications, data is therefore transformed between many different data structures “one by one”. Each “point to point” integration inter-dependes with at least two applications evolving independently along their individual upgrade paths. In contrast, today’s best practice is to leverage a “canonical”, meaning application-independent, data model, which acts as a hub for all integrations. This data model abstraction removes complexity from integrations.
- Last, but not least, complex integrations are implemented as programmatic procedure (e.g. PLSQL in case of E-Business Suite or SQR in the case of PeopleSoft), making them much harder to debug and maintain an integration logic that is implemented declaratively (e.g. using the visual BPEL process flow).

Validating and correcting such “brittle” integrations one by one after an upgrade can easily consume as much as half of your entire upgrade budget.

“Integration issues was the most cited reason for ERP replacement strategies”

— Aberdeen Group: Best Practices in Extending ERP

“The enterprise application silos were never designed to communicate freely with other applications that make up a typical business process”

— Aberdeen Group: Achieving more Value from Enterprise Applications

“Implicit” End to End Processes

Third, it’s ultimately end-to-end processes that deliver value to the customer, not individual enterprise functions. End-to-end processes typically span multiple packaged applications such as CRM, Supply Chain as well as Financials.

This said, today’s enterprise IT systems were not originally designed with end-to-end processes in mind. Processes today are nothing more than implicit constructs retrofitted from a combination of multiple packaged applications, each with their own customizations and held together by “brittle”, point-to-point data replications between the individual applications.

In practice, validating end-to-end processes as part of an upgrade can be a money sink. The reason is that a process has so many possible failure points. It will fail if only one of the multiple involved customizations or only one of the multiple involved interfaces breaks down after an upgrade. Furthermore, end-to-end processes are rarely documented explicitly, which makes them very hard to clearly define and manage. Imagine how much money you could end up spending validating an undocumented process.

HOW SOA-ENABLEMENT REDUCES UPGRADE COSTS

Now that we understand the downsides of traditionally used shortcuts, let’s take a closer look at how Service Oriented Architecture (SOA) can improve the situation. Due to significant benefits, the entire industry is moving towards SOA (Service Oriented Architecture). This means that best practices for integrating and customizing applications as well as automating end-to-end processes have fundamentally changed. We will demonstrate how applying the principles of SOA to customizing, integrating and building end-to-end processes with packaged applications will reduce costs and risks of application upgrades.

Essentially, SOA is an architectural style optimized for integration. Its base units are well-defined services (based on the web services standard) designed to be re-usable by other applications or processes. Because web services adhere to a universal standard, the integration that leverages a web service does not need to know about the specifics how this web service is technically implemented, e.g. what database table it accesses behind the scenes. The web service therefore helps to “de-couple” the outside world from the “guts” of the packaged application.

The concept of “de-coupling” (or “loosely coupling”) has significant ramifications for application upgrades. It helps us to build “non-intrusive” integrations, which do not reach into the application’s internal database, but leverage web services built on top of application business logic and therefore become “upgrade resilient”. Similarly, customizations and extensions “layered on top” of such web services will remove the trouble during upgrades known from customizations built with native application technology close to the application’s database schemas. Furthermore, creating extended functionality on top of existing application web services is the far

better choice than adding a further niche application to the mix and having to synchronize duplicate data stores. In this case, the SOA concept of “reuse” avoids the perils of integration shortcuts we have learned about earlier. Last, but not least, end-to-end processes can be easily built by orchestrating a series of web services. In fact, a standard business process orchestration language exists called BPEL (Business Process Execution Language) designed for this very purpose. Hence building your customization and integration strategy on web services will most certainly facilitate process-orientation.

Importantly, creating SOA together with packaged applications has now become mainstream. Whereas even a few years ago, most packaged applications didn't have well-designed interfaces with the right level of granularity, today all major ERP and CRM applications provide the necessary interfaces to be part of a SOA (Service Oriented Architecture). Very recently integrations required proprietary adapters and customizations were best implemented in the native application technology. Now, all Oracle Applications Unlimited product lines (Siebel as of Siebel 7.8, PeopleSoft as of PeopleTools 8.47 and JD Edwards Enterprise One as of Tools Release 8.97, Oracle E-Business Suite via the Oracle E-Business Suite Adapter) expose web services.

At the same time, Oracle SOA Suite provides an easy to use, comprehensive and industry-leading SOA toolset, which is pre-tested and certified with all Oracle Applications and interoperable with other middleware technologies. Oracle SOA Suite includes an Enterprise Service Bus (ESB) for routing and transformation of messages, a Processes Engine based on the BPEL (Business Process Execution Language) standard to orchestrate web services, Business Activity Monitoring (BAM) to monitor process activity in real time as well as Oracle Web Services Manager to manage security and performance of individual web services. In addition, application framework tools such as Oracle's ADF (Application Development Framework) and its WebCenter next-generation portal allow to easily build user interfaces on top of web services.

In fact, leveraging Oracle SOA Suite will provide a further benefit that reduces application upgrades: Because tools like BPEL, ESB, ADF, BI-Publisher etc. allow to create Integrations, UI and reports “declaratively”, meaning via visual tools and configuration of pre-existing building blocks rather than having to write programmatic code, your implementation will be much easier to “debug” during the upgrade validation phase.

Let's look in detail at how SOA principles help avoid the typical Enterprise Architecture shortcuts that experts have identified as key to expensive upgrades:

- Custom code required to adapt the off-the-shelf functionality
- Linkage/interfaces to third party systems
- End-user business processes that span multiple applications

“A successful approach is to develop customizations outside the framework of ERP, allowing the company to keep up to date with the latest release from its ERP vendor”

— Aberdeen Group: Best Practices in Extending ERP

Layered Extensions/ Composite Applications

Tailoring the functionality of packaged applications into customer-specific solutions will remain a necessity in the future for most customers given the vast portfolio of diverse business functionality. The key difference lies in the design of such “customizations”. Based on the SOA mantra of “loosely coupling”, the new best practice is to layer customizations/extensions on top of Application web services. Implemented “outside” the core application, these customizations/ extensions will be “decoupled” from the vendor-driven evolution of the packaged application through upgrade cycles. If you extend functionality this way, you will reduce upgrade costs based on the vendor commitment to provide upgrade-resilient application web services.

Architecturally, such extensions, also called composite applications as they can “compose” web services from multiple applications into one user interface, are built using an independent orchestration and user interface layer “outside” the applications. Oracle Fusion Middleware provides the necessary tools with SOA Suite’s BPEL Process Engine for the orchestration layer and ADF (Application Development Framework) as well as BAM (Business Activity Monitoring) and also BI Publisher (for reports) together with Oracle WebCenter for the user interface layer.

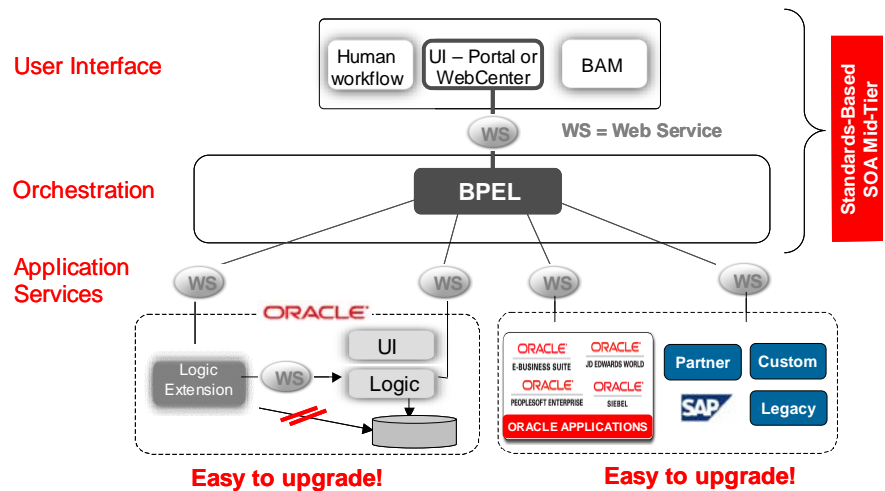


Figure 1: Composite Applications instead of Native Customizations

In contrast, purchasing a “best of breed” niche application to complement missing functionality would add yet another database with possibly redundant data and issues arising from the previously discussed data synchronization shortcuts. A good example is an e-commerce server purchased to provide online shopping services in addition to the existing order management functionality in the core CRM/ERP application portfolio. Following the layered extensions approach, the SOA best

practice would be to build an online store right on top of web services that expose existing order management capabilities from the packaged application. This approach promotes “re-use” of existing functionality and avoids having to synchronize duplicate data stores.

Both Universal Lighting (self-service extension for warranty management) and Viewsonic (special pricing approvals extension) have created such “non intrusive”, upgrade-resilient application extensions on top of Oracle E-Business Suite leveraging Oracle Portal and BPEL. Using such state-of-the-art tools allowed both to implement these solutions in 2-3 months. Similarly, Oracle Support delivers its own Metalink web support capabilities on top of Siebel web services leveraging ADF as well as Oracle Web Services Manager (OWSM) for security. Salem State College’s previous native PeopleSoft customizations took up to 4 months to validate during upgrades. Those were removed and necessary functionality re-implemented as a composite web application.

Note that many customizations are in fact reports, which could be implemented as custom legacy code (e.g. SQR in the case of PeopleSoft). Leveraging BI Publisher against XML data sources will reduce upgrade costs due to “declarative” report design and “decoupling” of source data and report layout. Similar to using style-sheets for web development, BI Publisher allows reporting templates to be applied to XML data sources. This is why Frontier Homes created its invoice check printing functionality leveraging BI Publisher together with JD Edwards Enterprise One.

Customer Success:

- **Universal Lighting: Warranty Mgmt. Self-service (Portal, BPEL, E-Business Suite)**
- **Viewsonic: Special Pricing Process (ADF, E-Business Suite)**
- **Oracle Support: Support Self Service (Siebel, ADF, Oracle Web Services Manager)**
- **Salem College: Customizations re-implemented outside PeopleSoft**
- **Frontier Homes: Check printing with BI Publisher, JD Edwards**

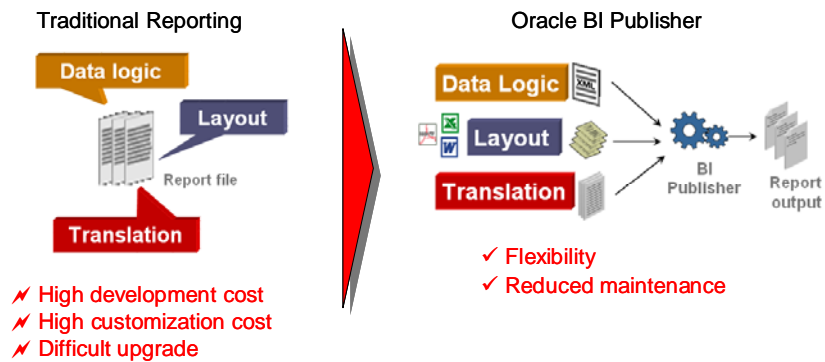


Figure 2: Benefits of Oracle BI Publisher

“SOA includes numerous industry standards,..., making the application-to-application plumbing less complex and risky”

— Aberdeen Group: Achieving more Value from Enterprise Applications

SOA-Based Integrations

If data synchronization, e.g. to a legacy system, is necessary, focus needs to be on creating “maintainable” integrations. In this conquest, well defined and “upgrade resilient” interfaces in the form of application web services are the first important building block. They offer stable integration interfaces that hide application technology concerns and also make use of validation logic built inside the application’s business logic.

In order to design integrations, and also processes, that are independent of application-specific interfaces, it is best practice to map to an intermediate data format (called “canonical model”). This can be accomplished by using the Oracle Enterprise Service Bus (ESB). In addition, Oracle AIA Foundation Pack offers a pre-built “canonical model” to use as the starting point of an SOA. BPEL (Business Process Execution Language) can then be used to implement error handling and more complex, multi-step integrations.

Note that the ESB as well as BPEL allow for integrations to be implemented “declaratively”, meaning that they are defined using visual tools rather than writing programming statements in languages like PLSQL in the case of E-Business Suite or SQR in the case of PeopleSoft. The benefits are significant improvements in being able to debug and modify existing integrations.

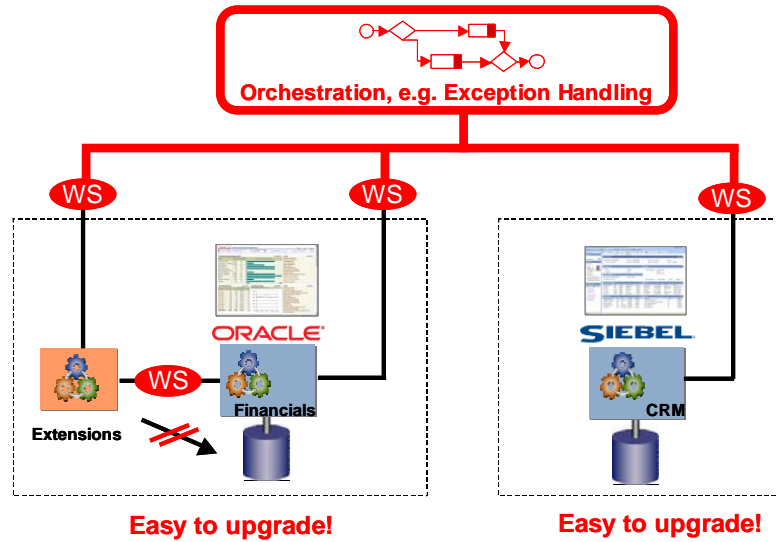


Figure 3: “Maintainable” Integrations

Customer Success:

- Large Bank: 2 h instead of 2 days to build integration SQR
- Helio: 4 instead of 8 months with SOA-based integration
- Fortune 100: 25% cost reduction for integration interfaces

Using SOA tools, customers have seen significant productivity improvements when implementing integrations. For example, a Financial Services customer running PeopleSoft experienced that a single interfaces previously built programmatically in two days using SQR could alternatively be created in two hours using Oracle ESB and BPEL. At a total of 300 interfaces, that makes a difference of over 500 man days implementing these integration interfaces while, at the same time, improving error and upgrade resilience. Helio, a virtual mobile network operator, completed

SOA-based integrations between E-Business Suite and an outsourced logistics provider in 4 months compared to estimates of 8 months using PLSQL. Another large industrial product company saved \$600k related to creating integration interfaces as part of a \$2.5M budget – a 25% saving!

Cross-Application Processes

“Making changes and updates to business processes are a lot easier in an SOA architecture than in a traditional point-to-point approach”

— Ted Cannie, Zanett

As we had explained earlier, end-to-end processes have, in the past, been made up “implicitly” of a combination of customized and extended application functionality together with interfaces to other systems. Hence, SOA best practices applied to the areas of customizations as well as integrations will deliver compound benefits in the context of end-to-end processes.

An important step is to explicitly document cross-application processes so that they can be efficiently tested without ambiguity during the upgrade process. Oracle provides a comprehensive toolset with Oracle BPA (Business Process Analysis) Suite for the business analyst’s process modeling tasks. Importantly, Oracle BPA features “closed-loop” integration with the BPEL engine, meaning that modeled processes can be easily executed with the BPEL engine and iterative improvements can be reflected back in Oracle BPA Suite.

Ideally, end-to-end processes are designed independently of the underlying applications. In this case, upgrading any of the applications that are part of the end-to-end process should have no or minimal impact. To create application-independent processes requires an intermediary “canonical service model”. Oracle’s AIA Foundation Pack (www.oracle.com/aia) provides such ready-made Enterprise Services to provide a layer of abstraction that is instrumental to designing end-to-end processes that are truly independent of the underlying applications.

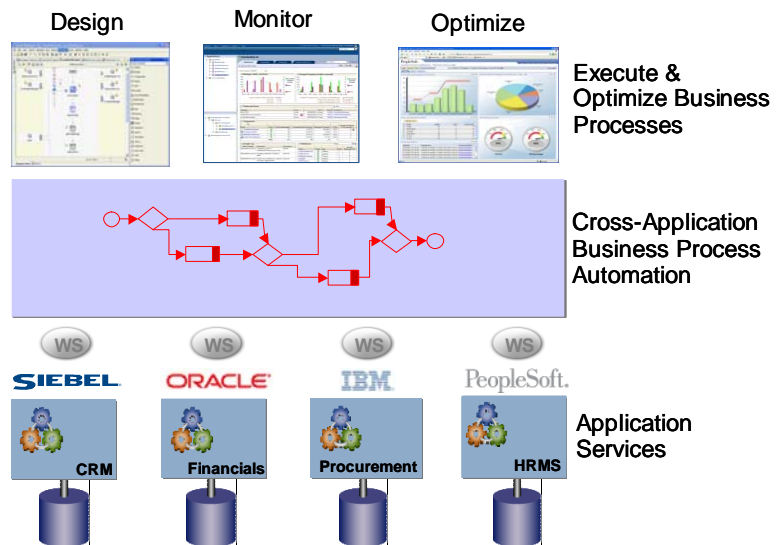


Figure 4: End-to-End Processes on Top of Application Web Services

Customer Success:

- **ABN Amro: BPM approach with PeopleSoft CRM**
- **Dallas Airport: On-boarding process in 7 weeks**

Financial Services Provider ABN Amro decided to leverage BPA Suite and BPEL to implement customer service processes that can span multiple interaction channels (e.g. call center, internet, branch office etc.) and will frequently be appended and changed when new offerings are introduced to customers. Before, these multi-channel customer service processes had been built via customizations of ABN Amro's PeopleSoft CRM application. The Business Process Management (BPM) approach much better supports rapid change cycles in the process itself while, at the same time, reducing the risk and costs of future upgrades.

Dallas Fortworth Airport, one of the USA's largest airports created a cross-application on-boarding process spanning PeopleSoft as well as E-Business Suite in only 7 weeks using Oracle BPEL. This BPM implementation replaces hard-to-maintain custom code and will be easier to validate and adapt during an upgrade.

NEXT STEPS

Ready to take steps towards reducing your upgrade costs? We have worked hard to reduce your learning curve. Customers tell us that Oracle SOA Suite is significantly easier to get started with than competitive products – in fact one customer compared a two day Oracle BPEL training class as more effective than a two week class for a competitive product. With a knowledge of XML, an essential survival skill for today's IT professional, BPEL skills will come natural. You can learn more about Oracle SOA Suite at <http://www.oracle.com/technologies/soa>

In addition, we have built best practice centers that specifically demonstrate the use of Oracle SOA Suite together with Oracle Applications:

- Siebel: www.oracle.com/technology/tech/fmw4apps/siebel
- PeopleSoft: www.oracle.com/technology/tech/fmw4apps/peoplesoft
- E-Business Suite www.oracle.com/technology/tech/fmw4apps/ebs

With end-to-end tutorials across SOA tools and Application technology stack, this "best practice" information will give you a "head start".

On this journey, keep in mind that success will come from a commitment to architecture. You will reap rewards from keeping customizations/extensions outside your core application, building your integration on top of web services that expose application business logic rather than synchronize from database to database and explicitly documenting end-to-end processes across your applications portfolio. With this commitment and Oracle SOA Suite's comprehensive toolset at your disposal, you can reduce the cost of your future upgrades



Application Upgrades and Service Oriented Architecture

April 2008

Author: Markus Zirn

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

Copyright © 2008, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.