Oracle Forms to SOA: A Case Study in Modernization

An Oracle Forms Community White Paper

Steven Price
Griffiths Waite
June 2008

ORACLE

Oracle Forms to SOA: A Case Study in Modernization

INTRODUCTION

Recent analyst reports and Oracle's own "Forms modernization" drive are encouraging customers to look closely at their technology investment to understand how it can be modernized and take advantage of changes in both business and I.T.

The key messages of this modernization approach are clear:

Retain Investment

With continued support and investment in the technologies of your legacy applications, you have the ability to modernize from a platform of stability.

Adopt Technologies

Service Oriented Architecture (SOA) allows you to realise new practices and technologies and to do so in a step-by-step basis

Integrate

Both your legacy and new systems can integrate and share services allowing legacy applications to continue to run your business as you build up new systems and services.

So, for many, the message of modernization is clear. The next step is to understand the practicalities and learn from the "hands-on" experiences of those who are already forging a path forward. We at Griffiths Waite have been involved in this area for a number of years and this paper presents a case study of a core business application built using Oracle Forms and its evolution from client/server, to the web and to a services based architecture.

Established in 1994, Griffiths Waite specialises in the deployment of SOA and business applications based on Oracle technology and Oracle applications.

This case study covers the implementation of an Application Processing System (APS) at two financial services organizations in the UK. APS is a credit granting system for consumer finance markets and processes finance requests from initial application through credit scoring and underwriting to quoting and compliance and finally to agreement completion. The first organization at which APS was implemented is one of the largest providers of financial services products to the non-standard consumer credit market in the UK operating with 5 call centres and a national network of over 450 branches and employing over 5,000 people. The

second company is a national loan broker with a network of branches throughout the UK and call centres in both the UK and India.

MODERNIZATION

Existing applications are the outcome of past capital investments. The value of the application investment tends to decline over time as the business and the technology context changes. Early in the life-cycle there will be enhancement investments to maintain close alignment with the business but eventually there will come a point where this becomes difficult. This can happen, for example, where the underpinning infrastructure is superseded, web access is required, or the weight of changes in the applications and lack of available know-how make it impossible to continue with enhancements.

Legacy Assets

Legacy systems are the result of many years of combined experience, development, and customisation, harbouring the core business processes that continue to provide competitive advantage. This combined with the reliability and stability of these systems in most instances makes a replacement plan poor value and high risk. Replacement and rewrite are necessary in certain instances, but if the existing legacy application meets current business needs, then the chances are that this legacy asset can be effectively transformed to continue to meet the needs of the business in the future.

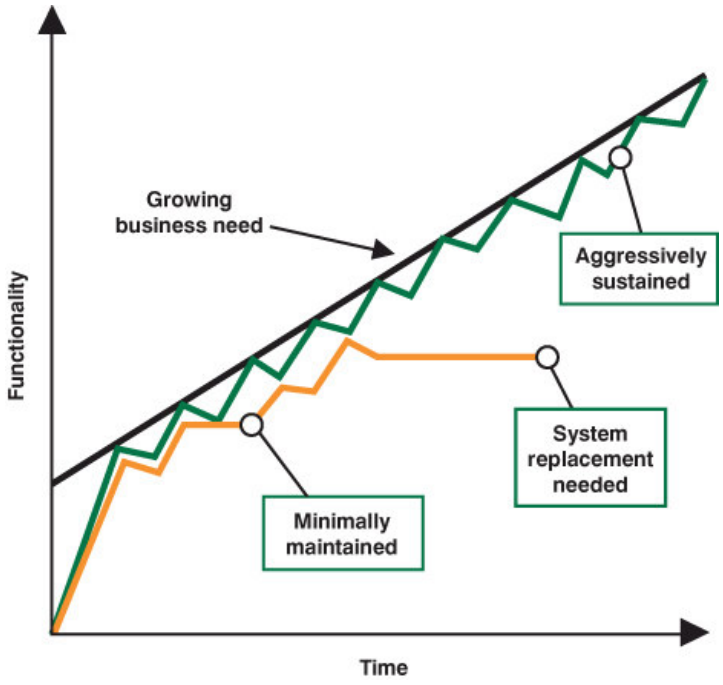

Figure 1 – Application Lifecycle

Active Sustainment

Preventing systems from slowly becoming legacy systems requires active sustainment. The goal is to prevent the system from "flat-lining" and consequently requiring a replacement effort. Sustainment lends itself to lower risks by avoiding the wholesale replacement of systems. Software sustainment is designed for the company's long-term health, not for short term effect. To avoid obsolescence, the sustainment must keep pace with changing technology and evolving business needs. Maintenance efforts should be punctuated with modernization projects, such as revamping an existing user interface, migrating to a new platform, or reengineering the code base.

Service Oriented Architecture

In a Service Oriented Architecture (SOA) systems are composed of reusable components, called 'services'. A service is a software building block that performs a distinct function — such as retrieving customer information from a database.

Wrap and Reuse

SOA takes standard business applications and breaks them down into individual business functions or services that can be used and reused to support different business activities. SOA works with legacy and existing applications through 'wrap and reuse' as opposed to 'rip and replace' – so services can be constructed, deployed and reused virtually on demand, and easily integrated enterprise wide, across multiple platforms.

Oracle Forms and SOA

The SOA Roadmap deconstructs legacy Oracle Forms systems into elemental components and recomposes them into a vast set of new application services.

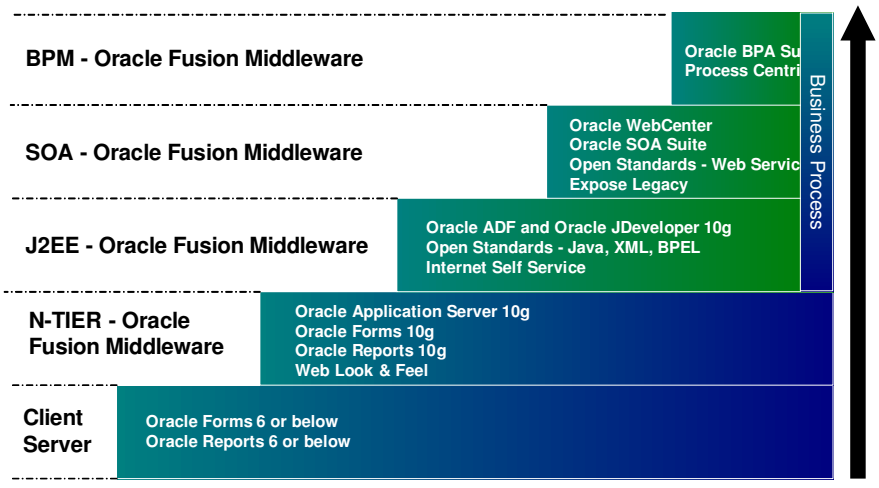

Figure 2 – The Forms to SOA Roadmap

In line with Oracle's own published roadmap, a step by step modernization approach limits risk and protects investment while taking a phased adoption of newer technological practices

The transition from Oracle Forms client/server to SOA should be a journey, not a single transformation. A staged (that is, phased) modernization effort enables Oracle Forms applications to be migrated over time. This lengthens the period of time during which Oracle Forms remains an architectural element, but reduces the overall migration risk during that time period. The first phase focuses on protecting customer's existing investments by stabilising the application and upgrading it onto a supported platform. Subsequent transformation phases will then gradually evolve the application to a Service Oriented Architecture.

The most critical and also the most difficult step in the transition to SOA is the migration to an n-tier Architecture focusing on establishing the business logic as independent to the database and client application. Once this has been achieved the transition to the upper levels of the Roadmap becomes significantly easier. Critically, system modules / business processes can be migrated one at a time and proceed at different speeds through the roadmap.

UPGRADING TO THE WEB

Client/server Forms

APS first went live in two call centers as an Oracle Forms client server solution. Following a successful implementation there was an urgent requirement to roll the application out nationwide to hundreds of branches, many quite small located in major towns and cities throughout the UK. These were linked to the HQ site by comparatively slow network links.

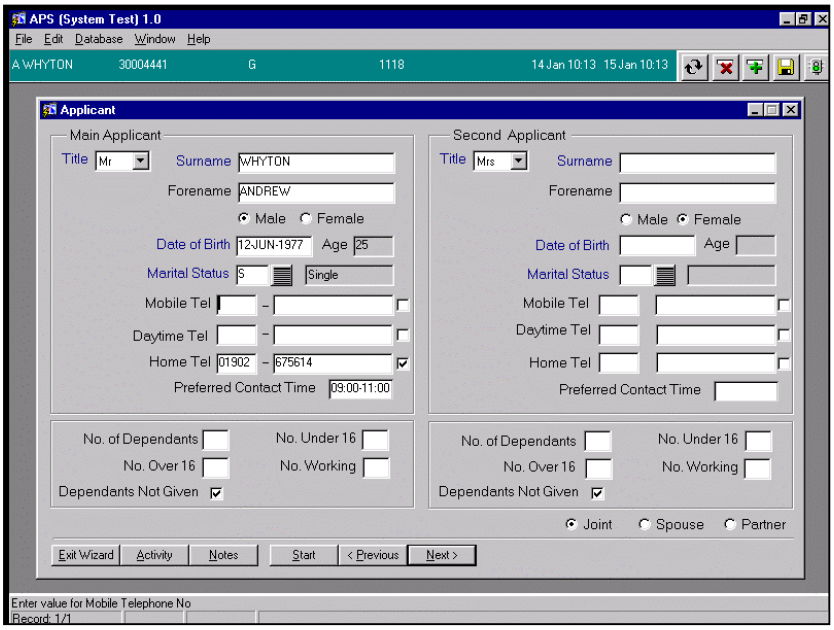

Figure 3 – Client/server APS

Running APS as a client/server application over a branch network was not feasible due to the performance problems of continually accessing the database across a

high latency network. Every operation a form makes (such as validating a field) often results in a number of requests and replies across the network between the form running on the user's PC and the database.

Web Forms

Upgrading from client server to web Forms was the most obvious first step to modernization. It brought a number of business benefits, put the customer on the latest and certified version of the technology stack, and provided a platform from which to take further modernization steps

The first step in the SOA Roadmap was the adoption of Web Forms to provide internet access to APS without making significant changes to the underlying platform. This is achieved by running APS on a Web server and emulating the graphical user interface in a generic applet in the client. Instead of the traditional Oracle Client Server Forms solution, a Java UI was presented to the user within a browser. The three tier architecture of Web Forms is much more responsive when run across a large company network, significantly reducing the network traffic between the client and the data center, compared to the client/server architecture.

Creating a Logical Middle Tier

Because the upgrade of Forms to the web was technically not difficult, it gave us scope to also make architectural changes to the application,

All business logic that resided in the client side forms was extracted and implemented as modularised code within the database. While web deployed Forms allows you the ability to run your forms in a three tier mode with little or no modification, we chose to refactor business logic from Forms in the database so that this processing could then be reused elsewhere in the application or even in other systems. A business tier was created within the database giving APS an N-tier architecture and allowing greater flexibility in implementing new business initiatives.

Equally important was establishing the foundation to leverage the integration features of Oracle's middleware and Java tools. Moving to an Internet deployment model also provided benefits in centralized management and deployment without having to migrate the technology stack or application.

New Forms UI using Java

Aside from the technical limitations in rolling out the client server version of APS to hundreds of branches of equal concern was how well the application would be received by the branch users. Like many Forms applications APS was fine in body but presented an old face to the world.

The dated user interface was perceived to be a major obstacle to implementing APS across the group and considerable resistance was anticipated. The system relied on the underlying database structure to drive its design, resulting in an interface where one window looked identical to the next and complex information was forced into numerous master-detail screens with cumbersome navigation.

We needed to rejuvenate APS's user interface. As user interfaces become richer in possibilities, more graphical, with more choices for both developers and the users, the importance of design increases. Software is a visual medium and in essence we wanted to give APS a 'facelift' with a new stylish design and attractive appearance.

Modernizing the UI gave the user immediate impact with virtually no risk in changing the application since nearly all the back end logic was unaffected by this change.

In redesigning APS's user interface our goal was not just to improve the 'look' of the application but also its 'feel' by designing an elegant solution that simplified the complex business processing. To this end we developed the concept of User Consoles. User Consoles are the primary focus for completing a particular activity. By bringing together all the related objects and tasks for a business activity User Consoles move away from the more primitive approach of table driven design and as a result they are easy-to-use and offer a compelling, intuitive experience for the user.

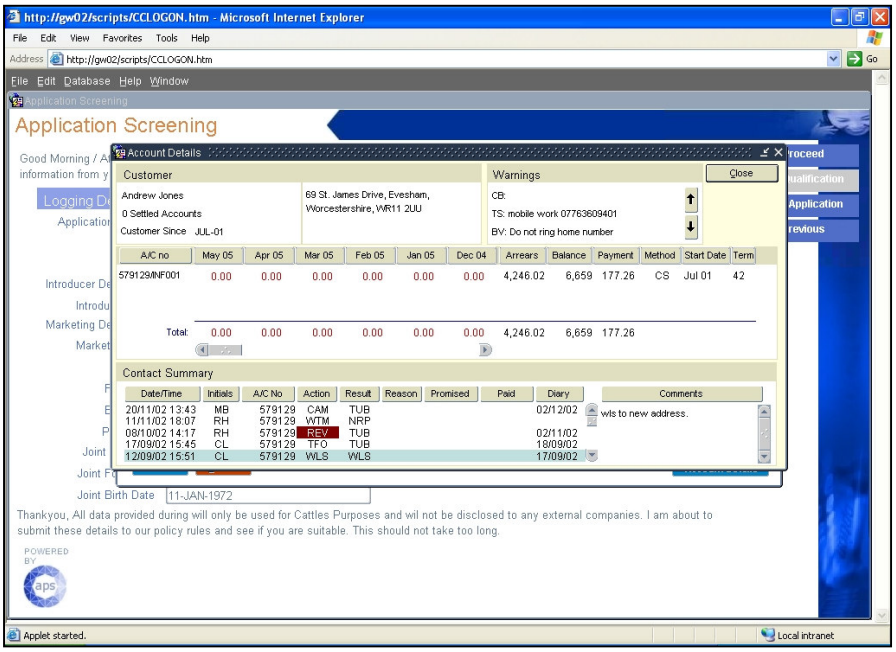

Figure 4 – Web Forms Java UI

Good design creates excitement and enthusiasm, and we wanted to engender a 'must have' mentality throughout the branch network that would lead to easier acceptance and adoption. To create the new 'look and feel' that we wanted for APS we took advantage of the Java extensions available in Web Forms to customise the Forms User Interface using JavaBeans and Pluggable Java Components (PJCs) to deliver a rich Web user interface.

Self Service Applications

While Oracle Forms fits many requirements of back office professional data entry applications, other tools and technologies are better suited for ad-hoc self-service kinds of applications.

Following the internal success of APS, there was a growing requirement to increase distribution channels, firstly, by engaging more closely with smaller brokers by offering them a service where they could submit on-line loan applications, and track their progress. This entailed providing these brokers with access to the same business processes performed by APS.

Many of the brokers were small companies and at this time they had limited Internet connections. This made the time taken to download JInitiator and the associated JAR files to run Web Forms limiting, and thus prohibitive.

These choices were made before Oracle ADF became established.

We needed another way to deliver APS functionality to this audience. After careful consideration, we chose the combination of Java Server Pages (JSP's) and the open source Struts framework together with a set of simple SQLJ-based classes for database access to build the APS Broker Portal. The main advantage of JSP is that the output is standard HTML and therefore requires little from the client except a compatible browser. There is no JVM running on the client, so there is no requirement for a set of Java runtime files or Java application files on the local PC.

The application ran within a web browser using a pure HTML user interface requiring no additional software to be installed or maintained. The web pages were dynamically generated using the Struts framework to control the rendering of JSP's. The Struts/JSP programs call upon the services of other Java components, some of which were implemented as Enterprise Java Beans (EJB's). These components avail themselves of the core business services provided by the APS PL/SQL stored procedures.

Sharing Services

The Broker Portal was developed using JDeveloper and deployed on the same application server as the Web Forms application allowing them to inter-operate and share the same infrastructure and common services. In addition, Forms and J2EE applications are able to share existing business logic by using database PL/SQL stored procedures.

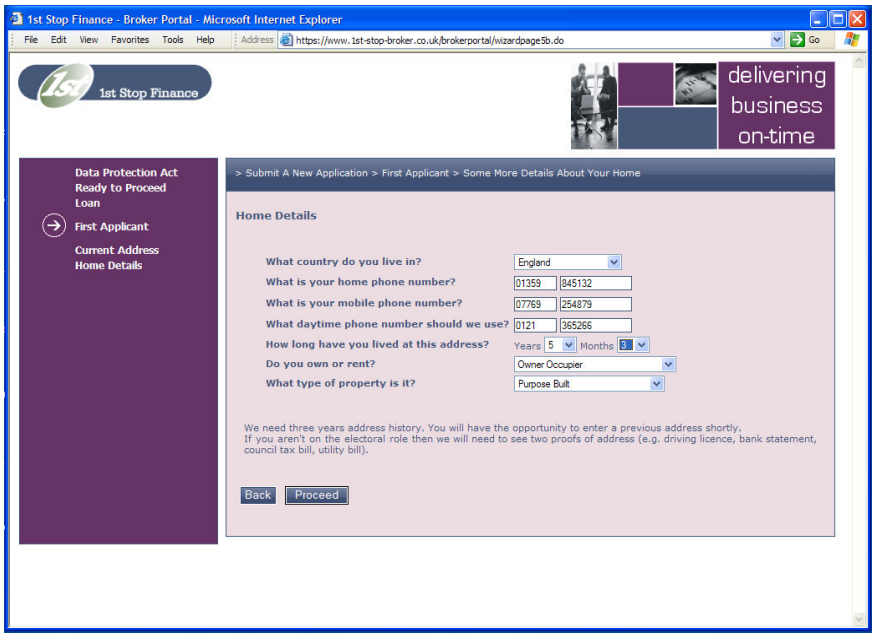

Figure 5 – Self Service APS – Broker Portal

APS's N-tier architecture from the Web Forms migration provided us with the basis for sharing the existing business logic with new business channels, greatly

reducing the time required to roll out this new style of business. This architecture enabled the maximum amount of reuse of existing code.

FIRST STEPS INTO A SERVICES WORLD

Following the publication of the lender's new five-year business plan a three year change programme was initiated. The first business initiative of the change programme – Electronic Leads – was commissioned to automate the application process to allow APS to accept large volumes of bulk-introduced business from 3rd party strategic partners.

Refactoring Forms Code to be Message Oriented

We decided to refactor the application processing into well-defined and self-contained services. Each business service was implemented as a self-contained group of PL/SQL procedures. There was to be clear interfaces to determine what data passed in and out of each service, with no direct interaction between the services. Instead when a service had finished, control was passed back to a single orchestration procedure that determined what happened next.

Orchestrating New Services from Forms

The architecture of the Forms Java client provides opportunities for the interaction between external programs or queues and Forms. We exploited this ability to write code that will listen for events coming from the Oracle Advanced Queuing (AQ) mechanism, which can then be passed through to the Forms runtime process.

The different application stages progress at different rates. Sometimes a stage may be temporarily unavailable. We needed some way to "buffer" data between stages and control its rate of flow. Oracle AQ (Advanced Queuing) enabled us to achieve this and significantly uplift the application volumes processed. Electronic Leads effectively implemented an Event Driven Architecture (EDA) using a message based Oracle AQ solution in the APS database. This significantly eased the transition to BPEL later.

FURTHER STEPS IN THE SOA WORLD

Business Process Management

To better support each partner's unique requirements GW embarked upon a Business Process Management (BPM) programme utilising Business Process Execution Language (BPEL), Business Activity Monitoring (BAM) and a Business Rule Management System (BRMS).

BPEL

The Electronic Leads module had given APS a loosely coupled process design. However the BPM supporting this architecture was embedded within the APS database. This put a reliance on APS for every link in the chain to be satisfied for

any given process to function. BPM should naturally reside outside any native system in the middle tier and be impervious to system change.

We wanted to move the process management and orchestration out of the APS database to the middle tier adopting an industry standard based architecture. To this end we selected BPEL, which provides a standard, portable language for orchestrating services into end-to-end business processes. It's built from the ground up around XML and Web services and is recognised as the industry standard with Oracle, IBM, Microsoft and BEA all heavily committed.

We implemented BPEL whereby the Electronic Leads processes where moved to the Application Server, exposed from the database as Web Services and coordinated through BPEL. The APS solution using BPEL was responsible for orchestrating and managing both the low level services within APS and the calls to external services and partner systems.

BPEL provided the developers with mix-and-match flexibility so that they only needed to revise individual APS components or plug in new ones as partner requirements changed ensuring that the Strategic Partner Programme could expand at an accelerating rate.

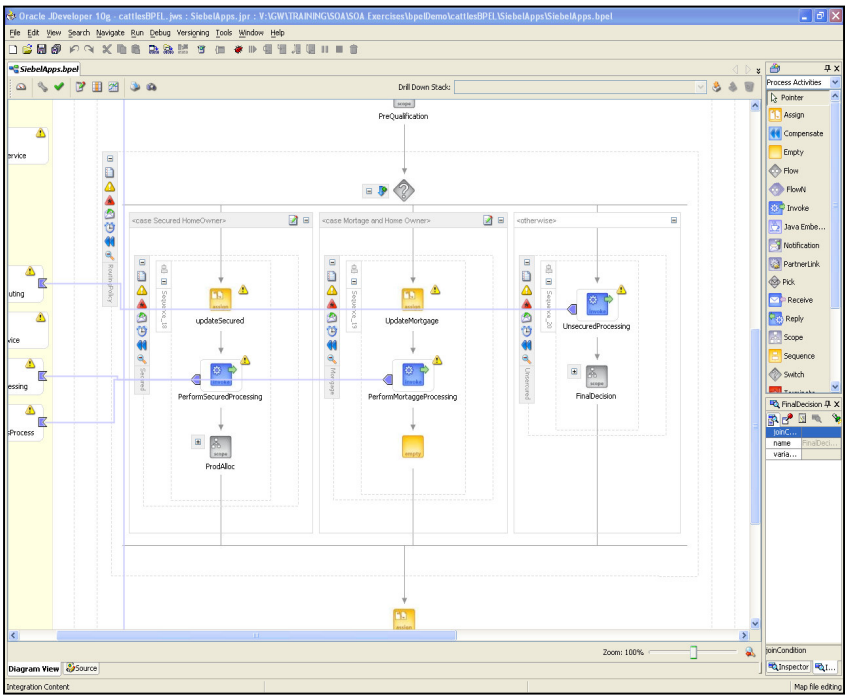

Figure 6 – BPEL Process Orchestration

BAM

To give immediate visibility into the Electronic Leads operation the Pipeline Manger was developed. Pipeline Manager is a graphical Business Activity

Monitoring (BAM) tool that allows the business to monitor and manage, in real-time, the process flows within Electronic Leads.

Oracle BAM enabled business users to define and monitor events and event patterns that occur throughout application processing. Sensors in the form of XML representations of events were added to the BPEL process to feed data to the BAM engine. These were then collated in BAM as reports. Users could view dashboards showing critical business measures including partner SLAs that updated in real time and then drill down into the detailed information behind them.

The Oracle BAM architecture delivers the requested information within seconds of an event or change in status. Because the primary source of data is messages, Oracle BAM can accept tens of thousands of updates per second into a memory-based persistent cache that is at the center of the Oracle BAM architecture.

The Pipeline Manager gives business users the detailed analytics they need to cut costs and improve processes – as business events are happening. This ensures that partner SLAs are met and customer service becomes more proactive.

Figure 7 – BAM Real-Time Analytics

Business Rules

To make APS more responsive to business change we extracted key business rules such as applicant pre-qualification and product allocation from the source code and built our own rule engine–Metis–to execute them. Metis is a Business Rule Management System (BRMS).

A BRMS allows users to change business rules in live systems without any assistance from IT. This approach recognises that business rules change frequently and need to be modified by business personnel. Putting rules development in the hands of business users avoids the long development and testing processes typical of traditional application development, enabling business users themselves to quickly and efficiently develop and modify rules as fast as the business requires.

The Metis rule development environment provided an intuitive, easy-to-use Rule Builder with a graphical user interface that enabled business users to rapidly design workflow, wizard and pre-qualification rules. The Rule Builder presented the users with meaningful business terms for data items and using an English-like syntax, complex rules could easily be created.

Metis generated PL/SQL packages from the rule definitions, which were then loaded into the APS database and made available as services to be consumed within the BPEL process. This resulted in a rule engine that was comparatively easy to build, and performant since the generated code ran inside the APS database and directly referenced the APS schema.

Metis enabled business users to customize APS without rewriting software - with dramatic results. In the six months prior to Metis' implementation we had performed one policy release comprising 200 business rules. Six months after Metis 30 releases had been deployed by which time the number of business rules in production had risen to 22,000. Metis enabled business users to 'get creative' and to start exploiting opportunities in the market whilst at the same time facilitating regulatory compliance.

A PLATFORM FOR GROWTH

Following the success of APS at the first client we were commissioned to implement the APS solution at a national loan broker. The company was experiencing rapid growth. This growth is both organic and acquisitive but the current systems environment was becoming a barrier to further growth.

The first APS module to be deployed was the existing Broker Portal application. This was deployed for the broker's partners and the group's UK branches and Indian call centre. Whilst the Broker Portal was initially deployed with minimal changes considerable new functionality was required to the Oracle Forms application and a second generation of the Broker Portal would later be required.

The client mandated that the solution be multi-channel, offer a compelling and superior user interface and provided management information that would enable them to drive the business forward.

Following the initial implementation of the Broker Portal we undertook a full review of APS's technology platform in light of the new requirements and now that it was scheduled to have a lifespan of a further 10 years.

Modern enterprise development increasingly entails working in a heterogeneous environment tied together by open standards. A business can be viewed as a series of services – both internal and external - that can be reused or orchestrated in various ways to build composite applications, cutting across traditional application and organisational boundaries. The focus within such applications is increasingly on assembling and orchestrating these services and building rich engaging user interfaces through which to access them.

Leveraging Existing Skills

APS was rapidly moving in this direction and we questioned whether Oracle Forms was still the best choice for satisfying APS's current and future needs. In moving forward we wanted to leverage our existing SQL and Java skills as much as possible and minimize the investment levels and time to productivity. Consideration was given to the available skill pools both internally and externally and the skill sets of the current and next generation of graduates. Whilst we were moving to an n-Tier architecture we wanted to reduce the number of technologies within our development environment as much as possible.

Oracle Forms is proprietary to Oracle and Oracle themselves have clearly stated that their long term technology strategy is based on the Java platform and whilst Forms can exist in a SOA world more modern next generation environments are better suited to the task. With a required lifespan of over 10 years Forms was not the best option.

Having discounted Oracle Forms we considered extending the 'self-service' application – broker portal. For partner and customer facing applications the user experience is critical. The main advantage of JSP pages—that they output lightweight HTML—is also the main disadvantage. Traditional HTML page-based Internet applications inhibit interaction in the interface; pages are slow to load, processes require multiple steps, simple functions such as scrolling down in a list of records, deleting a record, or changing the way information is sorted requires a refresh of the page and a small, standard set of controls limits the ability for developers to create truly engaging, user-friendly applications. These limitations do not allow for a dynamic user experience and lessen productivity, while causing frustration as users fail to complete tasks leading to process abandonment.

It is clear that the stale "click and wait" interaction associated with traditional web applications is no longer acceptable. Users now expect real-time updates and desktop-like functionality. Consequently, we needed a better way to mimic desktop functionality within the confines of the existing web application model.

Oracle Application Development Framework

We settled on Oracle Application Development Framework (ADF) and Oracle JDeveloper. JDeveloper had evolved to a 'Forms-like' integrated development environment (IDE) for building service-oriented applications using the latest industry standards for Java, XML, Web services and SQL and offered a single

platform to assemble rich UIs, develop business services and orchestrate those services.

Oracle ADF is an end-to-end application framework that builds on J2EE standards to simplify the development of service-oriented applications by implementing the Model-View-Controller (MVC) design pattern and further extending it by providing a model layer that can expose business services from disparate technologies. Separating applications into these layers simplifies maintenance and reuse of components across applications. The independence of each layer from the others results in a loosely coupled, Service Oriented Architecture. Furthermore, Oracle ADF lets developers choose the technologies they prefer to use when implementing each of the layers.

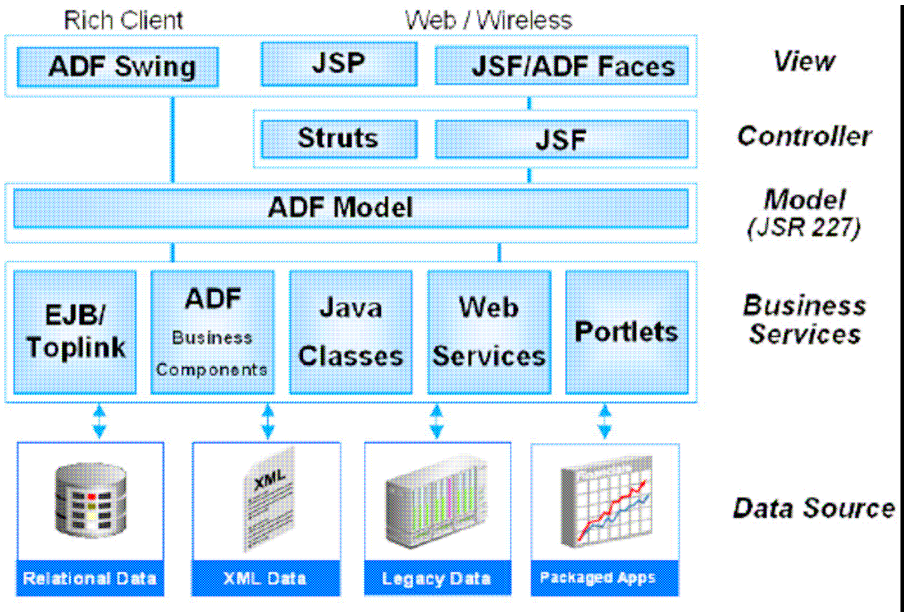

Figure 8 – Oracle ADF Technology Options

We adopted ADF Business Components as our primary technology for database interaction. Oracle ADF Business Components is a framework focused on creating objects, which implement the Business Services layer on top of a database. ADF Business Components enables CRUD style applications to be developed with productivity levels approaching those of Oracle Forms whilst enabling our programmers to work in a familiar SQL 'centric' development style.

A RICHER USER EXPERIENCE

UI Services

Just as we adopted BPEL for process orchestration we were keen to take a standards based approach to UI development. In selecting our UI technology we wanted to combine the ubiquity of the Web browser with the richness of user interface elements found in traditional desktop applications to enable us to create

truly interactive and engaging user experiences, ones that could run on multiple platforms with minimal or preferably no rework required.

Java Server Faces (JSF) is a component-based UI specification for building Web user interfaces in the Java EE 5 stack. JSF is fast becoming the industry standard for web UI development with IBM, Oracle, BEA, Sun and RedHat all committed to using JSF as their UI technology.

Within the view layer Oracle ADF includes a set of rich functionality JSF components – called ADF Faces which are built on top of the standard JSF APIs. Oracle ADF also enables AJAX development to be undertaken in a more productive declarative fashion than previously possible.

JSF further provides a highly flexible rendering architecture that defines a loose coupling between component behaviour and presentation. This allows developers to build client-agnostic UIs that cater to an assortment of output devices. The same programming model (JSF) can be used whether we want to build a web application, a portal application, a wireless application, or a dynamic and interactive AJAX user interface.

Retiring Oracle Forms

Following the selection of ADF we replaced the Oracle Forms with ADF Faces and any remaining client side code was refactored into PL/SQL stored procedures. At the same time the use of BPEL was expanded to orchestrate all of the processing within APS.

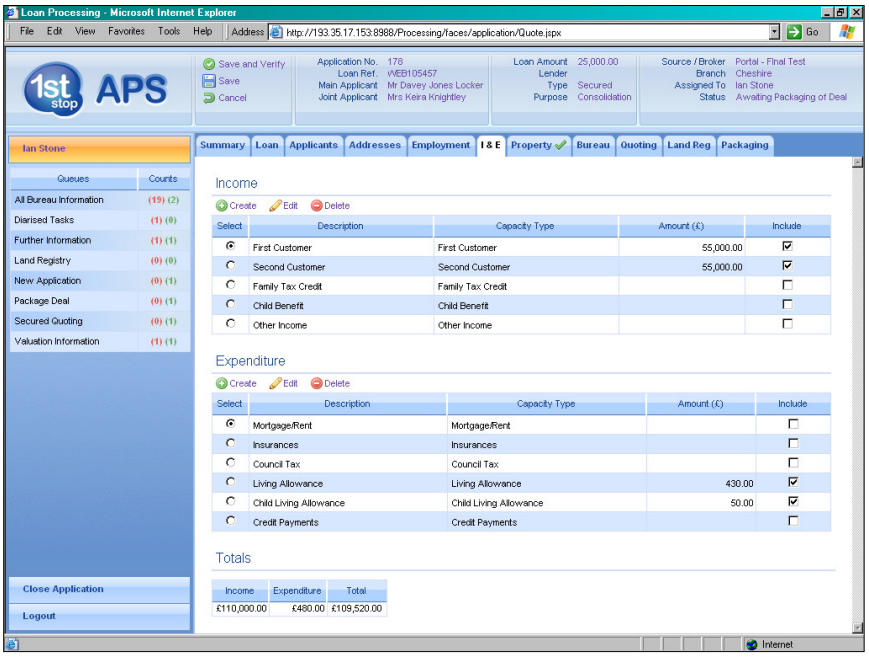

Figure 9 – APS ADF Faces

Replacing the JSP Solution

For the improved user experience that was mandated for introducers and customers we needed to restore the interactivity and usability that was lacking in the JSP solution and make APS's web experience more like the desktop. It was also important to simplify the online process as much as possible to ensure that customers didn't abandon the application process without finishing.

We therefore decided to build a new set of screens specifically for the Customer Portal. The key advantages that ADF afforded us was the ability to not only produce richer more attractive user interfaces but also to be able to reuse core validation and business logic from within the business services layer across different channels.

Figure 10 – Customer Portal ADF Faces

Decision Services with Oracle Business Rules

At the same time we migrated to Oracle ADF we adopted Oracle Business Rules in preference to Metis. Whilst, Metis delivered many of the benefits of a BRMS it suffered from a number of architectural limitations.

Oracle Business Rules runs as a completely separate decision service, employing its own meta-model over the corporate data model. This means that it is protected from changes to the APS data model, and more importantly, the business rules are now reusable across other business systems.

The approach which Oracle Business Rules takes to defining the rules is also very different from Metis. It is more declarative whereas our own approach was more procedural (for example, with Oracle's rule engine it doesn't matter what order you declare the rules in, so a latter rule can "fire" meaning that an earlier rule then becomes applicable – a process called "inference"). The big advantage of this is that Oracle's rule engine can optimise the rules, and thereby avoid evaluating rules which are no longer relevant.

Oracle Business Rules is supplied with a GUI Rule Author, but this is not really suitable for end users. We substituted this with a program which reads the metamodel and the rules from a collection of spreadsheets and then loads these into the rule repository using Oracle's SDK. This provides the users with a more approachable user interface, and also enables us to enforce elementary validation of the rule definitions.

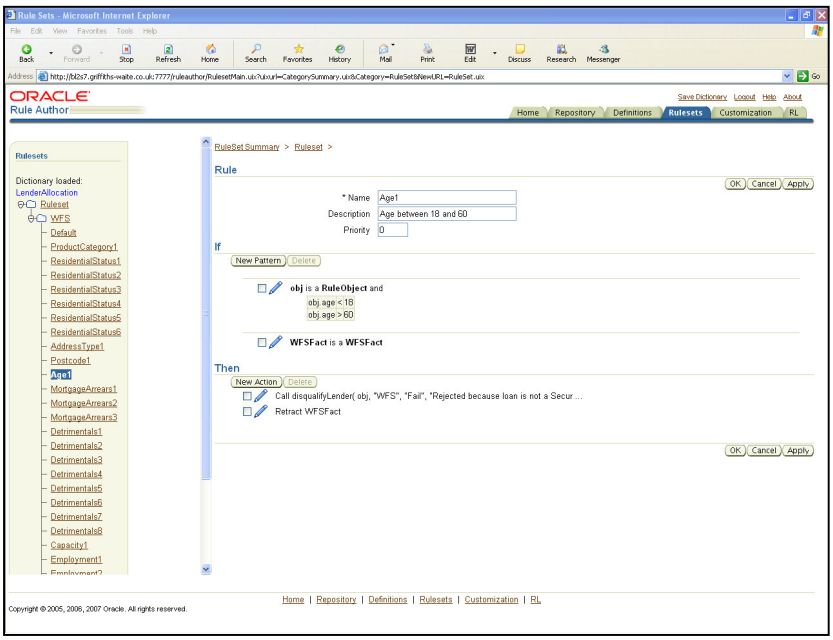

Figure 11 – Oracle Business Rules

SUMMARY

All software is susceptible to a natural decline in quality and value over time without continual, balanced investment in functions, features, architecture and technologies. We recognized that the nature of the applications we built was changing and SOA was the best way to meet these challenges.

The evolution from client/server to SOA took place over a number of years and in multiple stages. At every stage we were keen to leverage as much of the existing solution and our existing skill sets as possible to minimise delivery timescales and risk whilst maximising our return on investment.

Using a phased, architecture-driven approach, we were able to achieve both these objectives and reverse declining software quality. Application modernisation through SOA maximizes the investment in existing systems by building on the strengths of the past and combining them with the opportunities that modern technologies offer.

By definition legacy modernization is a journey which is never complete and we are now currently implementing Oracle BI into APS and deploying the Oracle BPA Suite to generate the process blueprints for BPEL.

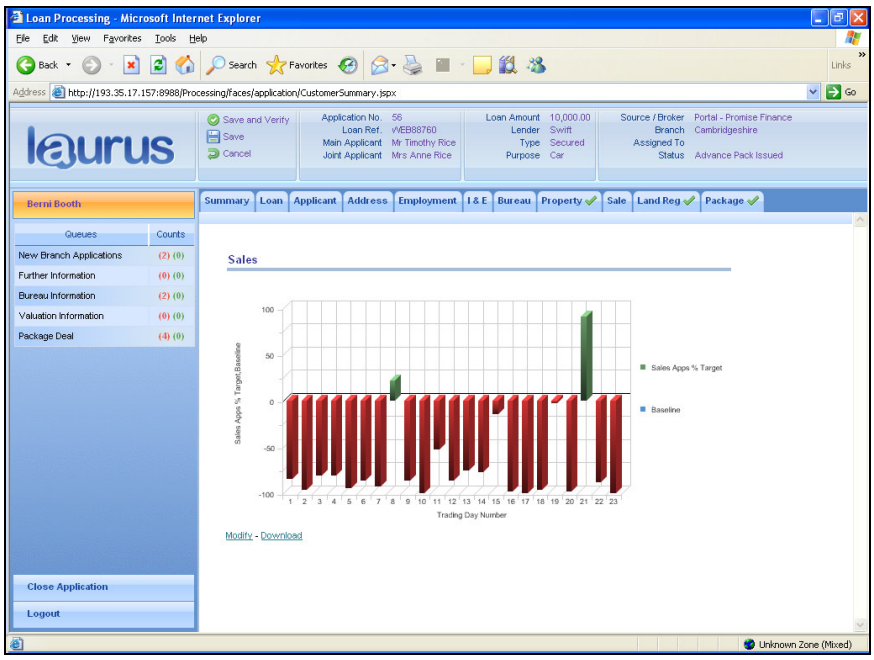

Figure 12 – Forthcoming Oracle ADF and BI Integration

Oracle Forms to SOA: A Case Study In Modernization
June 2008
Author: Steven Price

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
www.oracle.com