

(PITSS.GB)

Optimal migration and updating of Oracle Forms

PITSS is with her PITSS.CON the supplier of a powerful tool to migrate Oracle Forms applications to a newer version or a different operating system. PITSS.CON automates this process completely, which results in saving costs and time. This is what PITSS.CON is all about.

Oracle supporters will immediately agree: Oracle offers a wide specter of development possibilities and technologies. The techniques that Oracle offers are quite robust. And certainly where it comes to web applications Oracle is able to deliver. So, that's not the problem. The problem arises when the question pops up: What has to be done with the existing Oracle Forms applications when you change? Is it possible to take the 'old' applications along to a new J2EE environment? If so, what are the costs and the efforts you have to put into this? And also: Is the advantage of changing to a new technology worth the costs and the risks of rewriting Oracle Forms applications?

More possibilities

Of course, as usual with new software applications Oracle Forms also raises her 'predecessors' (the older versions of the applications, we mean) to apply with the new edition. But is that sufficient? Not often as the new application and technology offers you more possibilities. And if you have more possibilities, you want to use them. That is understandable.

In such cases Oracle often advises you to rewrite the application, to make it fit for a J2EE-architecture and if done so to integrate it, together with other modules, with the application server. Why? Because J2EE contains the promise that – by its use of open standards – on the long run the maintenance of the applications will be easier and the connectivity with external products will be simplified. So, is this then a story that ends with 'and they lived happily ever after'?

No, because many DBA's consider J2EE as 'complex' and 'difficult'.

Developers of Oracle Forms – mostly 4GL – are used to a high level of productivity, which is not possible in J2EE. That is the reason why Oracle advises the use of her Application development Framework (ADF) as an 'easy to use' tool for the implementing of J2EE-designs by creating a clear, strong visual surround ding that minimizes the need to rewrite codes.

But rewriting demands a thorough knowledge of the Forms application that needs to be rewritten. And often that is the stumbling block: the people that know the application, no longer works for the company or don't remember it anymore et cetera.

Rewriting also demands up-to-date knowledge of the latest technologies and developments if you want to hold on to the existing functions or – even better – if you aim to improve these for the new Forms.

All together it means that special attention is needed for planning and monitoring (cost control), which makes the need for project management for such a rewriting task unavoidable.

A palette of skills

To be successful with the rewriting job, you need a palette of skills. Skills like knowledge of PL/SQL, Forms and Java, Java Script, XML and also experience with J2EE are of essential importance. *Please note: we are not talking about 'or this or that!' We are talking about 'and this and that!'*

New user interfaces often offer better accessibility. Browser based applications have their limitations at this point. If it is not possible to adjust things to that, it might be wise to use the latest still supported release of Forms.

And then there are also different 'business logics' that need to work together in the same group of action codes. We are talking about user interface actions, data validation, data manipulation et cetera. If this code is changed to ADF, it will be necessary to split up the 'business logic' in components and redefine them based on the MVC design pattern if you want to create a real ADF architecture.

MVC

MVC means Model View Controller and is a design pattern that splits up the design of complex applications in three units with different responsibilities:

- * data model (model)
- * data presentation (view) and
- * application logic (controller).

To separate these responsibilities improves the readability and the reusability of the code. It also prevents for example that changes in the

The solution to all this is to translate all the codes to Java or to remain the PL/SQL code and place these in the database. There is also the possibility to choose for a mix: have the user interface in Java and the data manipulation in the database. This last named possibility is often recommended. That is where PITSS may play an important role!

PITSS namely combines the knowledge of the architecture of the Oracle Forms and Reports application and the analytical possibilities of PITSS.CON with the powerful reengineering power of the Application Analyses and Application Engineering modules and their Forms2ADF Assistant, Business Logic Assistant, Web Services Assistant.

PITSS.CON loads the Forms and Reports application in its repository, decomposes all code and rebuilds the complete structure of object interdependabilities. The thorough knowledge of the complete application makes it possible for PITSS.CON to maximize the number of components that can be changed to the ADF world in a 'natural' way.

Most of the Forms business logic deals with data. That this is part of an Oracle database goes without saying. PITSS.CON assists with the migration of the data intensive business logic to the database, where a Business Logic Layer is created.

After the code has migrated from Forms and Reports to the database, this layer may not only be addressed from Oracle Forms and Reports, but also from Oracle ADF, Oracle APEX and might even be used as web service, which enables entrance from another user interface surrounding. Whatever happens in the future with the user interface surrounding, you will therefore always be able to readjust this development without a real problem.

Business Logic

Business Logic is in ICT the collective noun for procedures that are agreed upon in a company. For example: an order is received, the system looks up in the Warehouse Management System (WMS) if the product that is asked for is available. If not it looks into the production planning, the customer is informed about the delivery

This 'parking' of the business logic is also a great solution for the common situation where the code is approached by different applications and it is necessary to keep some of these applications in Forms, while others demand ADF. In that case the business logic needs to be available for different interface worlds, while the common aspect for all these worlds is Oracle's database. This way the application will be suitable for Service Oriented Architecture (SOA). The whole SOA concept namely is based on the reuse and integration of components.

The rest of the applications contain mostly Forms components which will be repeated within the ADF MVC pattern by using ADF Faces, JSF (Java Server Faces) and ADF Business Components.

The process in 5 steps

In a white paper named 'From oracle Forms to Oracle ADF and J2EE' (2009) Magdalena Serban and Bahar Us explain how this works.

It starts with the loading of all the applications and their database schema definitions into the repository of PITSS.CON. A detailed analysis will be carried out then. In this phase the strategy that will be followed is planned. This is based on the specifications of the application which are: The complexity of the application, the demands that the application requires considering the user interface and the degree in which the application needs to be adjusted to the web model.

This *first* step of the process is called **the Application Analysis**.

The *second* step is titled **the Dead Code Analysis**.

In this step the application will be stripped of all unnecessary code. Experience learns that there is 20 up to 30 percent 'pollution' in the way of unused tables, libraries, duplicated and/or unused code units and even completely unused functionalities such as a calendar for a data field which is no longer necessary in a J2EE surrounding.

Step *three* contains **the migration of the business logic to the database**.

Herewith the base is laid for a successful step to a Service Oriented Architecture (SOA). Time and budget can be saved by using tools like:

* **BL Assistant**

The BL assistant extracts services from Forms and Reports by just pushing a button. The program constructs the chain of codes needed for building a service and visualizes the degree of usability for migration to the database. It then executes the migration, during which the recorded code for the initial application is renewed. De so created database services are not only available for every Forms application, but also for ADF applications and may even be used as web services.

* **Web Service Wizard**

This tool translates all recorded functionality to a web service and by doing so it makes the application suitable for the outer world.

* **DAL Assistant**

DAL means Data Access Layer. This tool modernizes the application which is based on tables. It makes it possible that it will be stored in line with the procedures. This is done by creating a Data Access Layer in the database and by using the application as basis for this layer. Later on this layer might be strengthened by the implementation of complex business rules and for more safety.

* **Application Analysis and Application Impact**

With this tool you will be able to map the impact of the proposed changes not only fast, but also accurate. It also enables you to identify redundant or problems causing code and to streamline the transition to a Service Oriented Architecture.

That brings us to the step *four* of the process.

This step is about **the reengineering of applications** within JDeveloper ADF with the help of tools. After getting rid of most of the business logic from the application, the remake of the users interface will cost you as much effort as pushing a button. The Forms2ADF Assistant automatically produces needed ADF objects such as:

- * Business services with ADF business components
- * Data access objects that may be entered from several ADF applications to support the easier read or write of business logic from or in an ADF application.
- * The finishing touch for the model, the view and the controller layers, during which each component of the Forms application (screens et cetera) will be analyzed and comparable ADF objects will be created. While doing this each object will be assessed for its individual properties and its possibilities of reproducing it in an ADF environment.
- * The user interface Java methods will be taken along when translating the PL/SQL: business logic to Java. By doing this an automated support will be created for variable re-mapping and for the calling of recorded business logic via data access objects.

Now we have reached step *five*, the final step of this process.

This step is called **Post-Generation Changes**. In fact this is nothing less than taking care of the necessary fine-tuning afterwards. The results of the four steps we set before, was well documented. That means that developers are blessed with a rich resource of knowledge and information which enables them to work with the converted application and also makes it possible for them to make a valuable estimate of the fine-tuning that still is needed.

That fine-tuning for example is needed for such things as:

- * fine-tuning the web layout
- * the redefining of the page navigation with the help of the existing components
- * the fine-tuning and enabling the tool translated Java methods
- * defining the components for which a tools supported recreation still is not possible or for looking up ways how to deal with the limitations of the web model (such as the access to a client file system or registry).

According to the authors of the white paper, this approach, with the help of PITSS.CON, is the best and most complete program for the task that we are talking about in this article.

Benefits of this approach and the use of PITSS.CON are the effectiveness, the productivity, the consistency, the natural architecture, the result which is free of software specific components, the support for developers that is available and the limitation of the risks that one takes by these kind of actions.

It might be wise to take a look at the website of PITSS (www.pitss.com). There you will find all kind of information among which a film of 14 minutes about the 'Forms to Forms Upgrade': <http://pitss.com/solutions/pitsscon-solutions/forms-to-forms/>

Sincerely yours,

Pierre van Daalen
IT-Staffing – OC Centor

Optimal migration and updating of Oracle Forms / March 2010

Authors: Pierre van Daalen, IT-Staffing OC Centor / Contributing Authors: Andreas Gaede, Magdalena Serban, Bahar Us / Review: Daniel Tin