Migrating an Oracle Forms application from 6i to 10g-Technical considerations

Hafed Benteftifa
December 2008
www.degenio.com

Abstract

The building group of Ville de Montreal, Quebec, Canada, decided to move its Building management system from the Oracle Forms/reports 6i to the 10g version before the end of 2008. This project was initiated based on several considerations, the most important one being the end of support for Forms 6i by Oracle in 2008.

The project was carried out in several phases which we will outline in the following paper. A number of issues were faced by the development team. While the overall project was technically challenging, there were a few issues that potentially could have derailed the project.

We will illustrate each case and explain the solutions we adopted.

I. Introduction

The building management system used by the Ville de Montreal is a system that was based on Oracle technology from the very beginning. Oracle Forms and reports were used to develop the application over a number of years.

The application itself is comprised of a number of sub-systems with interaction between the various systems as found in classic ERP systems. Oracle Designer was used to generate some of the sub-systems although most of the links between the form modules and the designer repository were broken due to maintenance carried using Forms builder. The system was deployed with the available technology of the time, namely Client/server technology with around 1200 forms modules and close to 600 reports modules.

While the system proved adequate during its expected shelf life, subsequent maintenance work introduced several non-standard artefacts, additional coding units and other issues that had the potential of rendering the application hard to manage for the coming years.

The developer base within the building group has expertise with Oracle Forms and reports developers. External consultants with other skills such as Java, database administration, etc… provide support for on-going maintenance work as well as development work for new projects.

An attempt to migrate the system to Oracle Forms and reports 10g was performed around 2006 using the Oracle migration tool (OFMA comes standard with forms developer) with hand coding the required changes between the 6i and 10g versions. About 40% of the modules were migrated at the time but the project was not brought to completion due to other pressing issues.

In the first quarter of 2008, a decision was made to carry out the migration from 6i to 10g before the end of year 2008. Both the database and applications were to move to the 10g platform.

The same team that was tasked to maintain the building management system, with the addition of 2 other resources, was to carry out the migration.

Skills set

At the beginning of the project, there were 9 people involved. The breakdown in terms of skills was:

Expertise	Resources
Oracle Forms	8
Oracle Reports	8
JAVA	3

Common to most Oracle shops and because of the existing Forms expertise, it was rather obvious that Oracle JDeveloper was not an option and the only available migration path for the building management application was to use Oracle Forms 10g.

II. Preliminary planning

Once the decision to use Oracle Forms 10g as the development tool for the new version was made, a preliminary document[1] detailing the project phases was prepared. A document that proved very useful at this stage was written by Paul Dirksen[2].

The document detailed the following steps:

- Adapting the obsolete features in the existing forms modules
- Changing the existing development workflow and adapting the developer workstations
- Organizing the deployment workflow on the new IBM AIX machine
- Developing an internal migration tool
- Inventorying the steps required for adapting the modules to the web requirements

[1] Guide de conversion oracle forms 10g, Hafed Benteftifa, www.degenio.com

[2] A Procedural Guide for Upgrading Forms6i Applications to Forms10g, Paul Dirksen, Digital Visions Enterprise Unit

- Inventorying potential problems with Forms 10g

Since our developers were expected to become familiar with the new Forms 10g features, a document[3] related to the new features in Forms 10g was made available to the developers. A document that details best practices[4] proved also to be very useful.

Features that are new to 10g were highlighted but most important were the features that were deprecated and potentially could cause problems after the migration.

Migration

When the first attempt at migration was done in 2006, changes in the modules were carried out manually whenever required. As such, it was near to impossible to adapt the modules to the new 10g requirements given the number of potential changes. A rough calculation showed that each module in the system required about 30 changes in properties with some modules requiring more than 100 changes.

Based on the fact that there were more than 1200 form modules, it was rather obvious that migration was to be done automatically. The Oracle migration tool, available with Forms 10g, could potentially migrate some of the modules. However, there were too many customizations that could be either lost of completely bypassed by the tool.

In addition, acquiring an external tool was not in the equation due to some budget constraints. As such, one of the developers was mandated with writing the migration tool either from scratch or using the oracle migration tool as a base.

Migration Tool

Based on ideas exposed by Wilfred van der Deijl[5] (Oratransplant) and Verschuren[6] (AMIS), the JDAPI API available with Forms 10g was evaluated. A prototype was developed based on code available in the builder help as well as the code provided on Wilfred's blog. We code-named this tool MouliForms[7].

Preliminary testing showed that the tool could be expanded to fulfill some of the migration requirements. The developer was given about 160 hours to deliver a working tool that would be installed on each of the developer workstations.

The features that are now available with MouliForms are listed as follows:

- Migration of forms from 6i to 10g. Close to 90% automated migration.

[3] Guide_bonnes_pratiques_forms_10g, Hafed Benteftifa, www.degenio.com
[4] Top tips for web-deployed forms, Peter Koletzke, Quovera
[5] Custom-built Forms Migration Assistant, Wilfred van der Deijl, see www.oratransplant.nl/2005/05/30/custom-built-forms-migration-assistant/
[6] PracticalJDAPIpaper, Harm H Verschuren, AMIS Services BV
[7] Guide_utilisateur_mouliforms, Hafed Benteftifa, www.degenio.com

- All the new features implemented with WebUtil are either flagged or switched at the code level.
- Search and replace capability: uses the JAVA Regex package.
- Batch mode: compile, replace, migration.
- Report calls modified at the code level (RUN_PRODUCT to RUN_REPORT_OBJECT).
- Attaching and detaching PLL libraries
- Forms compare: before and after in a tree like fashion. Useful to see what has changed between versions.
- Full Reports for outputs: HTML, CSV, PDF, Excel summary reports.
- Use of Log4j for various developer level reports (debugging-info etc.)
- Full JAVA Swing interface with switchable French-English interface.

There are still some bugs mainly related to the interface but Mouliforms does the job and makes full use of the JDAPI package.

MouliForms was heavily used during the migration phase and is now used for searching (objects, strings) and locating specific module calls as a replacement to the search facility found in Forms builder.

III. Functionalities

Printing

One of the main differences between the client/server and web deployments lies in the printing capabilities available to end users.

Our requirements were diverse and required a fair amount of tweaking.

- Some reports need to be outputted in duplex mode.
- A number of reports are automatically printed at the end of the week and the end of each month.
- Format for most reports is 81/2x11 but some are in 81/2x14.
- All of our reports are in French and hence most of them do contain characters with accents.
- All of our printers are network ready and support Postscript level 2 and 3.
- Some of our reports are printed on special paper, for example Avery label paper.

Based on those considerations, the migration from a Windows platform, where all our special printing was handled, to an AIX-based platform was not as smooth as we expected.

We did encounter specific issues with:

- Setting up Arial and Courier Fonts
- Setting up Italic since it was printed back slanted
- Setting up duplex printing
- Setting up printer PPD files

In order to solve some of the font problems, a number of modifications were carried out in the report instance configuration files. The steps we followed are:

- Converting fonts from Windows to AIX.
- Setting up the configuration information for these fonts in reports, mainly by modifying entries in uifont.ali. This enabled proper support for PDF output.
- Setting up the ppd files for each printer and the required format considerations
- Enabling report.sh to use a special version of lprint in order to allow the output of French accented characters.

These were not the only changes we made in handling our printing requirements. Since all reports were now being processed on the reports server, we modified the way our Forms modules made use of reports.

In this case, a procedure that handles all reports calls was developed, based in part on the previous migration attempt source code and code found in Steve Callan's document[8]. The various printing requirements (Excel, mail, RTF and PDF) were heavily tested given the number of reports available with our application.

Email

Some of our reports are emailed to recipients based on some specific requirements. Some of these are listed here.

- The report needs to be attached to the email.
- Multiple recipients might be selected
- A message might be added to the email

Without the message requirement, we could have used the email capability of reports server. However, because it was required to have a message added to the email, it was deemed necessary to use a plugin from the reports server plugin exchange website. In this case, we used BLOBdestination.

Graphics

There was a number of oracle graphics 6.0 already used in some of our existing form modules. While there was a way to install graphics on 10g, another approach was advised and that was use BI beans. These are available with the Forms demos on the OTN website.

[8] Installation Cookbooks: Deploying Reports on the Web, Steve Callan

The setup was straightforward and the changes that were introduced were mainly in the formsweb.cfg file.

Client-Side Utilities

A number of client-side functionalities are present in our application and was to be maintained in the new 10g version. Given that the Forms module are now server deployed and as such the old code, with reference to client access, is now referring to server drive locations, we converted our calls to WebUtil Calls.

WebUtil is provided by Oracle and the setup is straightforward on the application server. The changes in the code were automatically performed by Mouliforms. In some very few cases, manual changes were done by developers.

Some small annoyances were reported by our users when browsing folders containing large number of sub-folders. The root cause was discovered to be associated with the way some of the Java methods in Webutil were programmed (use of Swing instead of AWT).

Corrections were made using a custom-built file selection javabean[9].

IV. Conclusion

Overall, the migration did go well as planned. However, there were several obstacles along the way. In such migration projects, coordination with the *infrastructure group* is essential given the fair amount of tweaking required on the application server.

The migration of a Forms application has to be planned based on a number of requirements, the most important ones being the deployment platform and printing usage aspects.

The benefits that we have already seen and reported by our users are a much quicker response time and the potential for expanding the application with PJCs and javabeans.

[9] File Selection Dialog with Oracle Forms 10g (10.1.2): replacing some of webutil functionalities, Hafed Benteftifa, December 2008, www.degenio.com