ORACLE® TOOLS DIRECTION

Migrating Oracle Forms to Fusion: Myth or Magic Bullet?

By Grant Ronald

Unless you've been vacationing on Mars or some other far flung and remote corner of the solar system, as an Oracle technologist you've heard of "Fusion": Oracle's strategy of next generation business applications based on the Java platform, SOA and Web 2.0. Contrary to what you may have thought, Fusion, or more specifically the core technology at the heart of Fusion already walks amongst us in the form of the latest version of the Oracle Application Development Framework (ADF). So, whether you see this new frontier as frightening or exciting, it is reality and it is here today. Which leaves one obvious question. What do you do with your existing technology? Technology that has been serving you well for years, and still continues to serve you well today.

The case for the future of Oracle Forms and Oracle's strategy for modernizing Oracle Forms is already well-documented http://otn.oracle.com/goto/formsmodernize and it's probably prudent not to waste a limited word count on re-emphasising the fact that there is nothing wrong with staying with Forms or having a blend of technologies. However, one question raises its head every so often: can—and probably more importantly, should—a Forms application be migrated to another technology like Java or APEX? This article takes a look at Forms migration, notes the challenges involved, the questions you should be asking, and whether there is a magic bullet for Forms migration or is it just a myth.

FORMS TO FUSION

It's probably fair to say that the challenge of migrating a Forms application is comparable whether we are talking about migrating to Java or a proprietary technology like APEX. Regardless of the programming language in your target platform, there are different concepts between the source and the target and much of the migration effort is trying map the two. However, given Oracle's strategic technology choice for Fusion is the Java platform and this question is something Oracle has had to address in its own Applications Division, lets focus on the challenge of migrating Forms to the Fusion technology stack.

And I'm doing this, why?

Before embarking on a project of migration, you might need to set some expectations. I think we can all agree that there will be risk, cost, and effort involved and so it would be reasonable to expect that what you get as a result of migration should in some way improve the business.

You may be considering migration because your business has fewer Forms programmers. Although, would it be cheaper to train PL/SQL developers to use Forms rather than migrate? You might be migrating because you want to avoid incurring an Oracle license to run Forms. But, are you just pushing your licence costs to another tier and would any potential saving be eroded by the cost of migration? Or are you looking to consolidate on a modern standards based technology; in which case maybe a more gradual evolution might reduce your risk while allowing you to ramp up skills.

And what are your expectations of a migrated application? Do you want it to look and behave like Forms, which gives your end users no visible improvements? Or so you want to get a more modern Web look and feel, which may involve having to retrain your users?

The bottom line is it's all about bottom line. Migration will cost you and you have to be very clear on the return you expect for the effort. For Oracle, the Fusion initiative had brought together from many acquired companies with a variety of solutions and technologies: some complementing, some overlapping and some competing. So for Oracle the chance to draw a line under this smorgasbord of applications and tools and move forward on a standard set of technologies was a major factor in considering migration.

The scope of the challenge

The first thing is to establish the scope of work involved. While on the surface, a Forms and a Fusion application may perform a similar business function, the "style" of the application is quite different. Oracle Forms has its roots in transactional client/server applications that are tightly aligned with the database. Conversely, Fusion Web applications are based on stateless Web technology and modelled around a service based architecture where each service is, typically, associated to a business process. And herein lies the first challenge of migration; any attempt to migrate a Forms application to Fusion without taking into consideration the difference in architecture is in effect an effort at re-implementing the 20-year-old Forms runtime.

Closely related to application "style" is the user interface. Web 2.0 and our "away from work" social-computing experience of applications are changing our perception of how these applications should look and function. Social networking sites like Facebook or MySpace, along with more commercial offerings like eBay and Amazon are a world away from the Forms multi-row block style of user experience. Users have more sophisticated expectations of applications than ten years ago and as a result are more demanding of the applications they use at work. And here lies the second challenge of migration: to re-architect the front end to provide a 21st century experience rather than just delivering an HTML version of the Forms UI. As an example, Figure 1 on the next page contrasts two new Fusion screenshots (foreground) with a Forms Implementation. Figure 1: Forms Building Blocks.

The third challenge is an interesting combination of the first two. Oracle Forms applications are built as a homogenised "lump" of business and UI logic. On the other hand, Fusion applications are based on Oracle ADF, which implements an architectural pattern called model-view-controller (MVC). This means that the implementation of a business service is completely separate from the user interaction

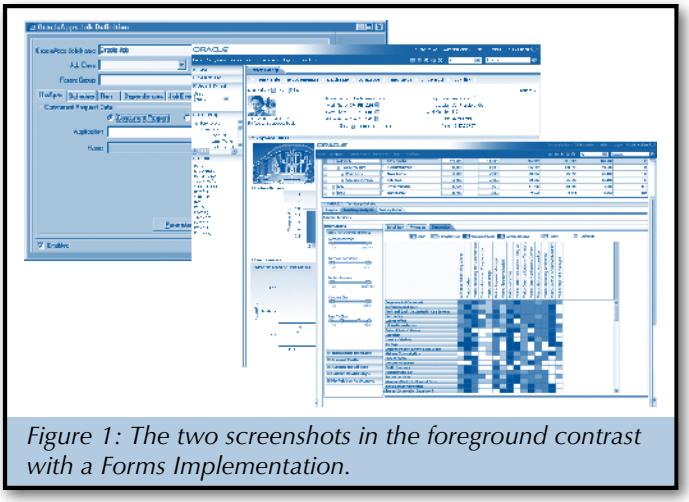

Figure 1: The two screenshots in the foreground contrast with a Forms Implementation.

code. For any migrated application to exploit the benefits of MVC the source form must somehow be refactored into UI and business/data logic. Which raises a number of interesting questions, such as: would the migration of a *when-new-record-instance* sit in the *view*, since you are navigating between UI components or within the model as you are selecting the next row of data. Or what about a call to *Next_Item*?

So the lesson from the three points above is: re-architecture. Any migration of a Forms application needs to take into account the target platform and for most options considered today, that means making fundamental changes to the structure of the application to better align it with the "sweet spot" of your target platform. And remember, that "sweet" spot isn't just the Java language, but the whole SOA world it supports. Do you really want to blindly re-implement your end-of-day-employee-updates-to-batch process or do you instead take that back in-line and use the asynchronous features exposed through BPEL? Do you want to mimic your custom-written expenses authorization process as-is or instead push that functionality out of your core application and into a rules engine and gain the agility and visibility offered by that solution? If you are not asking these questions then you are not modernizing.

Maintaining for who?

The next challenge of migration relates to you developer community. Oracle's own Fusion development community features developers from database, PL/SQL, Forms, People Tools, .NET, and Java backgrounds. So, do you migrate your application to look familiar to your Forms developers or your Java developers?

The simple answer is you need it to be maintainable by a Fusion developer regardless of their background. Simply re-implementing Forms structures and built-ins as Java classes and methods suggests, at best, only a cursory attempt at re-architecure and should set alarm bells ringing. Furthermore, a 3GL Java implementation seems like a step back from the productive and declarative nature of Oracle Forms. And this is where Oracle ADF brings together the two worlds. Based on Java EE design patterns and best practices, it fits comfortably with Java architects, but with concepts familiar in Oracle Forms and driven by metadata, it provides the rapidly productive and declarative experience that a Forms developer would expect. And this is a major reason why Fusion uses Oracle ADF and why any attempt at Forms migration is more likely to succeed if based on Oracle ADF.

Are you speaking my language?

Closely related to the developer community is the development scripting language. Forms is based on PL/SQL, the Java EE platform is based on Java. So, you need to migrate all your PL/SQL code to Java? Not necessarily all, but definitely a considerable amount of it. PL/SQL is without match for what it does: manipulating data in the database. If you blindly migrate all your PL/SQL to Java you will, in all probability, end up with less maintainable and less efficient code. Instead you should consider which code it best left as PL/SQL. In many cases, you will be able to identify PL/SQL code in your Forms application that is doing straight database interaction and often that code will sit more comfortably in the database as PL/SQL. Not only is it probably more efficient and maintainable but also it will certainly be less effort to put it there than try to redevelop it in Java. Of course, doing that still requires refactoring of the code so that references to Forms built-ins or items are removed.

The next step is to identify Forms code that supports features already implemented in Oracle ADF. For example page navigation, validation and lookups are all implemented in PL/SQL in Forms. However, Oracle ADF already provides that functionality as part of the framework and doesn't require Java code to implement these features. Migrating that PL/SQL to Java doesn't make sense given the functionality is already implemented in a way that allows much easier maintenance and customization.

Which leaves the rest of the code. The above tasks are anything but trivial but migrating the rest of the Foms code is almost certainly a more considerable effort and the bulk of your undertaking. As pointed out before, to blindly migrate PL/SQL code to Java, regardless if you have some clever framework to map the code, suggests that little consideration is given to the requirement to rearchitect. Careful consideration has to be given to how that code fits in the new environment. Of course, the final complication is that in Oracle Forms the flow of code executed as a result of an action is not necessarily contained in once place. Therefore, a particular action of, for example, submitting a new employee record may result in many Forms triggers firing. Each of these discrete actions may themselves call trigger code, program units, attached libraries or even calling our to third party code such as OLE or C/C++. Understanding the flow of these triggers, and mapping these to the multiple event points in your new technology is one of the most difficult challenges.

CAN YOU ACHIEVE MIGRATION?

The final question really comes down to how to achieve a migration. Within Oracle's own Application Development Tools division the decision whether to attempt migration was taken early on. As Christopher Job (Vice President of Tools)

commented in his foreword to the book *Oracle JDeveloper for Forms and PL/SQL Developers* – Koletzke & Mills:

"One of the controversial decisions we tool took early on was not to tackle the task of migrating Forms and PL/SQL applications to J2EE. We did not think that in the long term, it was the right thing for our customer, and consequently, it was not the right thing for Oracle."

Christophe continues:

"We knew we would not be able to provide a complete migration, and manual modifications would have been required for all but the simplest application. [Furthermore] the applications resulting from any automated migration would not have had the structure of an architecturally sound J2EE application."

Thus, Oracle made the decision that for reasons including, but not confined to architectural differences, the need to exploit new business and technology concepts and the scale of the challenge for an enterprise application meant that a modularised and phased manual redevelopment approach was the only real option.

And so, Oracle's Fusion applications are embarking on a process of staged manual development of Forms modules, the development of which is aided and accelerated by using Oracle ADF and reusing existing database PL/SQL.

So, are there any tools that can help? Well, some companies have innovative solutions, some of which are an aid to refactoring, some offer a "kick-start" by converting some of the Forms metadata to target platform metadata, and others advertise more complete migration.

Each of these offerings needs to be evaluated in the light of those points made above and expectations set accordingly.

IN SUMMARY

The path you are likely to take is influenced by the place you want to be. If your motivation for considerating a migration to a different technology is to exploit the features and power of that target then you need to be willing to make wide ranging architecture changes. To consider migration without rearchitecture is handicapping your target technology to work in way for which it was never designed.

So, Forms migration, myth or magic bullet? Can you take migrate a Forms application to a technology like Java? Of course you can, but is it the right thing to do and can it be shown to be a success? That is a much more difficult challenge. Oracle's experience in this field is that the difference in technology, and the goal to fully exploit the new technology means redevelopment rather than migration is the choice that fits best with the business. After all, if Oracle could have cast the die for that magic bullet, wouldn't they have done it by now?

WHAT ABOUT APEX?

While JDeveloper and Oracle ADF remains Oracle's strategic choice for enterprise application development, Application Express (APEX) is another option for Oracle developers and the recent 3.2 release offers a Forms converter. So, has the APEX team found the secret to this migration magic bullet?

Not surprisingly, Forms to APEX migration has to address pretty much all of the same challenges. As noted previously, and also in the Application Express documentation, there still remains a fundamental architectural and functional mismatch between Oracle Forms and APEX (stateless vs. stateful, UI design, and business logic replication). So, in all but the simplest of emp/dept examples, considerable re-architecture decisions have to be made. While the Forms to APEX converter maps some of the structures within Forms (such as blocks, LOVs, item properties) to the to APEX equivalents; and allows you to annotate components to help in the post generation phase, re-architecture decisions still need to be taken and redevelopment is required for all Forms triggers, the code in those triggers, Forms program units, PL/SQL libraries (*.pll), menus (*.mmb) and object libraries (*.olb).

About The Author

Grant Ronald is a group product manager working for Oracle's Application Development Tools group responsible for Forms and JDeveloper where he has a focus on opening up the Java platform to Oracle's current install base. Grant joined Oracle in 1997, working in Oracle support, where he headed up the Forms/Reports/Discoverer team responsible for the support of the local Oracle Support Centres throughout Europe, Middle East, and Africa. Prior to Oracle, Grant worked in various development roles at EDS Defense. Grant has a BSc. in computing science and has been working in the IT industry since 1989.