



Aqualogic BPM

How to step over the fence and make ALBPM to be a killer BPM ?

Glauco Reis (glauco@portalbpm.com.br) is a BPM and SOA consultant. He works for more than 25 years in IT, and has more than 150 published articles in Brazilian magazines (www.glaucoreis.com.br). He is the Chief Editor of PortalBPM, a magazine with more than 6500 readers.

Oracle BPM suite (previously BEA and Fuego) is one of the most impressive and advanced BPMS solutions available on the market. A very good chunk of visual edition, just a few code needed and one of the most BPMN compliant are closed in the same box .

Despite all these ahead of vision, there are some faults that must be fixed to be a killer BPMS. When I say faults here I not mean the “bugs” that every solution’s market already has.

These are architecture faults instead, and some are so serious that even the simpler implementation bring headache to developer.

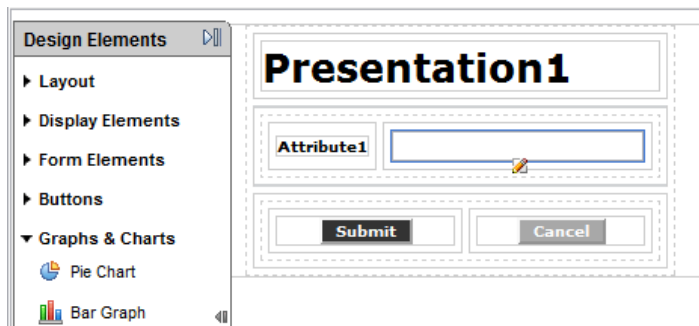
That’s what we are going to explore in this article, and how Oracle could bypass those problems.

User interaction

Fuego started as a “human centric” BPMS solution, wich means they focus on orchestration of people interactions. People communicate their intent through “screens” (Picture Q1). Usually the presentations are created visually using the development tool (studio), or we can import JSPs and use most of the java code around the world.

There are some programatically screens created through script, not recommended by Oracle and not part of our study here. Let’s explore the problems related to JSP or presentations usage :

Picture Q1. Sample of presentation (Oracle BPM 10)



Regards JSP usage

1 – Only “BPMObject” types can be sent to JSP files. It’s a big problem, since there is a variety of code types we desire to use inside JSP, and JSP can handle any JavaBean like code as

Published 2009/05/23



www.portalbpm.com.br

well. Place this limit for BPMObject seems to be a non sense implementation. The tool must be fixed to use any JavaBean compliant code. Far more strange it appear to be, BPMObject itself is a kind of JavaBeans implementation nowadays.

Workaround : We can use inheritance to make a BPMObject compliant with other existing java code.

2 – Only one BPMObject can be sent to a JSP file. This is far worst than before problem, and we can see in the Picture Q2 this limitation. We are enforced to create a BPMObject with the JSP appearance, wich is against all we do in nowadays implementations of web pages. Today we create a page with composites (smaler pieces of code like tags, scripts, portlets) to make a complex page. BPMObjects are enforcing exactly the oposite (join the complexity in just one code piece of code).

workaround : There is no workaround for this, and what we usually do is create a BPMObject with the same name of a JSP page. Inside it we put other objects using inheritance to reuse other BPMObjects´ behaviours.

This not fix the problem, since just the behaviours are reused. All data instances can not be used between pages. The data must to be copied along BPMObjects on the process flow. I really think Oracle could redesign the page to have more than one parameter to be sent, improving reuse along the pages.

Picture Q2. Information sent to a JSP

Implementation Type
BPM Object Interactive Call

Select BPM Object Variable: var1

Use BPM Object Presentation

Use JSP Presentation: jsp_screen.jsp

Input Display

3 – There is no JSP editor inside the tool. We can find several JSP or XML editors over the web. Oracle could put one of them as part of its product.

workaround : Well, since there is a Eclipse behind the Oracle studio, you can install a plugin by yourself (just take care of Eclipse´s version).

4 - JSPs can´t be used outside Screenflows. This is a weird behaviour, since the presentation can be used both in Process or Screenflows.

The main differences from ScreenFlow and Process are :

- Screenflow are harnessed to a single Role, meanwhile a process across along the participants. In really we can see a ScreenFlow like a roll of screens for a single participant.
- ScreenFlow are stateless regards its execution, and process are statefull instead. In a simple way : If power has over, a ScreenFlow´s execution starts from the beginning, while a process instance continue from the last execution.

Workaround : Oracle recomends user´s interactions to be made as Screenflows. This is the best practice we must follow on, since it can be made the only option in the future. What is not clear is how to use Screenflow in transaction systems, since there is no state on it.



Regards presentations usage

5 – Presentation editor is limited. The version 6 had a very limited version of a editor for presentation. The screen is handled like a matrix, with XY positions for each component. It's not possible to drag a component to a filled position. This editor were improved in version 10, thanks god.

Workaround : Migrate to version 10, if you can. There is no other editors available on internet, since the file format is proprietary.

6 – No JavaScript code for presentation elements. Presentation is a XML file, with a very few resources. It's not possible to create new components, nor even use AJAX and javaScript. There is no migration tool between presentation and JSP as well. In other words, if you used presentation and need to migrate to JSP, every screen must be created.

Workaround : I really hope Oracle spend some time creating a “round trip” tool for conversion of presentations to JSPs.

7 – Presentations are closed inside BPMObject. Another weird thing here. A presentation is something part of a BPMObject. If you delete the BPMObject, it's presentations will be removed as well (it's a kind of composition). We can define several presentations inside a BPMObject, but otherwise is not possible. This is completely against any Web pattern, because a Web Page is made by compositions of small pieces of code (Tags, Scriptlets, portlets) to make the role page. There is no way to use the same BPMObject in two diferent presentations.

Workaround : From my point of view presentations must be independent from BPMObjects.

8 – No roles for screen fields. It's a common task to see the same page in several roles. But for each role, there is a diferent set of fields that are going to be viewed, and some fields are read only for a role and editable for another one. In the Oracle implementation, a presentation is completely against any role.

Workaround : JSP can bypass this limitation. We can put a field with the participant role inside BPMObject, and a scriptlet or taglib could use this information to present or not the information for user.

BPMObjects

BPMObjects are like the business components of ALBPM architecture. They carry data between the activities, and the fields can be used on the gateways to make decisions. As XML files are so easy to make diferent versions, and they become JavaBeans at deploy time. It's far easier to create a BPMObject than a Java file, even for a business analyst.

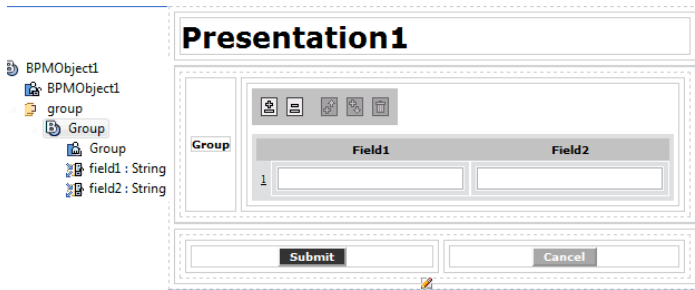
BPMObjects usage

9 – Hierarchy of just one level. If you desire to make a collection of fields, you can use a group. It's far easier to make visual screen for group (Picture Q3). By one hand is easy to create a visual collection of objects, and by another hand there is only one level for a group.

Picture Q3. Visualisation of Groups



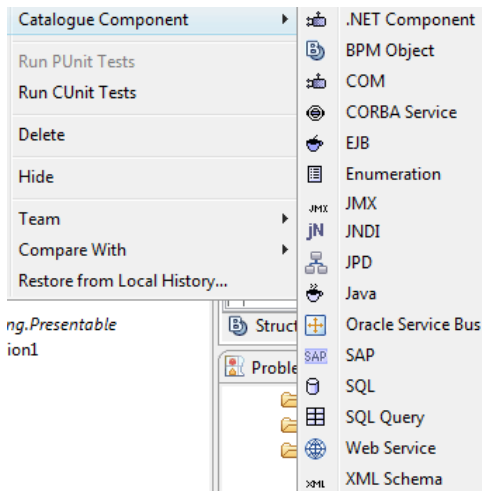
www.portalbpm.com.br



Integration

Legacy can be reached because ALBPM has a catalog of components. The catalog can be made from several component types (see below), and for WebServices we just need to point WebServices' endpoint. Once imported, several XMLBeans code are going to be generated, and we instantiate through scriptlet the XMLObject specialization classes.

Picture Q4. Catalog of Component



Usage of services

9 – Services and screens not easy to connect. Lets imagine the steps for a single interaction. At first a screen is created with edit fields. Once activated, the screen values are collected and sent to Web Services, wich generate new values for presentations. The problem is that screen only understand BPMObjects, meanwhile Web Services just understand XMLObjects (XMLBeans). To get informations from screen and send it to WebServices obligate scripts like the following :

```
for each XMLelement in BPMObject.element do
  element as ObjectType.Element = ObjectType.Element()
  element.field1 = XMLelement.field1
  element.field2 = XMLelement.field2
  element.field3 = XMLelement.field3
```



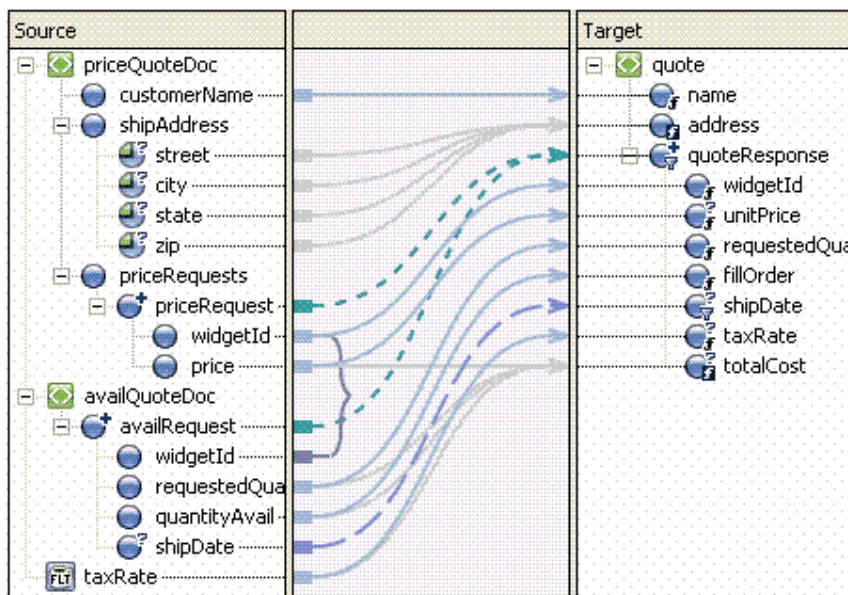
www.portalbpm.com.br

```
element.field4 = XMLelement.field4
element.field5 = XMLelement.field5
sub1 as SubType.Sub1 = SubType.Sub1()
sub1.field1 = XMLelement.field1
sub1.field2 = XMLelement.field2
element.sub = sub1
```

Believe-me : There is a lot of situations like this on a real program. The work may be so hard, and a copy field by field is needed. A single field misspelled can make the program instable.

Workaround : Oracle could make WLI Mapper as part of ALBPM tool. Instead of scripts for this copy, it could be created visually.

Picture Q5. WLI Mapper



Development X Production

Oracle BPM studio is an integrated development tool. Visual editor share the same space as a integrated internal TOMCAT version. Once tested, it can be published as EXP files (it's a ZIP file with another extension).

10 – Production version is not the same as the development one. The TOMCAT internal version had a lot of features cutted. User does not log on workspace, for example. If you need to use LDAP informations to make a decision is not possible. Most of code using PAPI are limited. There is no full log as production version does.

11 – There is no visual debugger. Debug inside ALBPM today is like the we usually did many years ago, I mean debug messages shown in console. There is no break point and variable analysers. Most of exception are captured and a new one is launched.

Workaround : I guess there is a long work to be made here. In the same way Oracle created visual debugger for ALSB (wich is made mainly by XML code), the same could be made in ALBPM.



www.portalbpm.com.br

Conclusion

Well, ALBPM is a very advanced, and now under Oracle umbrella we probably are going to see a very fast improvement of this tool. As we told before, ALBPM is in the middle of a full programatic tool and totally visual ones, were no code is allowed but the limits are well defined. Fix some of its limitations will make a unbeatable tool. In the next years the market will need more and more tools that make development cicle shorter, and perhaps BPMS solutions are cloeser for this goal.

Links

<http://albpmsupport.bea.com/documentation>

<http://albpmsupport.bea.com/download>

The mais source for ALBPM code and documentation