



www.portalbpm.com.br

# Aqualogic BPM

## Como as limitações do ALBPM podem ser resolvidas para que o produto se torne matador ?

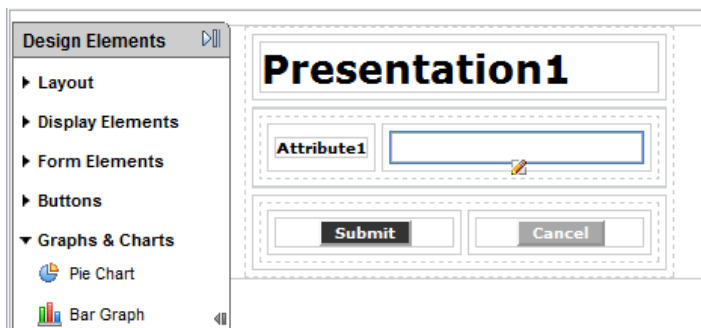
**Glauco Reis** ([glauco@portalbpm.com.br](mailto:glauco@portalbpm.com.br)) é consultor BPM e SOA. Atua há mais de 25 anos em TI, com mais de 150 artigos publicados em diversas revistas nacionais, além de participação em eventos como COMDEX e FENASOFT. É especialista em SOA e BPM, tendo atuado na construção de uma solução BPMS nacional, articulista do site PortalBPM ([www.portalbpm.com.br](http://www.portalbpm.com.br)). Também mantém um site pessoal explorando os temas OO, UML, SOA e BPM ([www.glaucoreis.com.br](http://www.glaucoreis.com.br)). É especialista em soluções IBM, com mais de 5000 horas de treinamento ministrado sobre SOA, BPM, Java, OO, UML e tecnologias relacionadas.

A solução Oracle BPM suite (previamente da BEA e anteriormente Fuego) é uma das mais avançadas soluções BPM (BPMS) do mercado. Além de agregar uma boa dose de edição visual, demandando pouco código, também é uma das mais aderentes ao BPMN. Embora seja uma solução muito avançada, ela têm algumas falhas que precisam ser resolvidas para que se torne “matadora”. Quando digo falhas aqui não estou me referindo a “bugs”, que naturalmente todas as soluções têm. Mas sim falhas na arquitetura, que reduzem a produtividade do desenvolvedor e tornam algumas atividades que deveriam ser fáceis em imensas dores de cabeça. Iremos mostrar neste artigo uma coletânea de melhorias e como a Oracle poderia focar neste processo de melhoria.

### Interação com o usuário

O Fuego se originou como um BPMS “human centric”, o que indica que seu foco é integração entre pessoas. As formas mais comuns para criação de telas para integração entre pessoas é utilizando JSPs ou “presentations” (Figura Q1). As “presentations” são telas criadas visualmente dentro da própria ferramenta, ou podemos importar e utilizar JSPs comuns, e aproveitar todo o arsenal de códigos JAVA. Também existem telas programáticas, que podem ser criadas via script, mas que não são recomendadas pelo fabricante. Vamos explorar agora os problemas relacionados à criação de telas utilizando JSP ou presentations :

**Figura Q1.** Exemplo de presentation





www.portalbpm.com.br

## Relativos à utilização de JSPs

### 1 - Somente objetos do tipo “BPMObject” podem ser passados como parâmetros para JSPs.

Isto é um grande problema, pois em projetos temos a necessidade de utilizar quaisquer tipos de objetos Java em JSPs, e estas por sua vez são capazes de gerenciar qualquer objeto do tipo JavaBean. Permitir apenas um tipo de objeto passado como parâmetro é uma limitação sem sentido. A ferramenta deveria permitir qualquer objeto com getters e setters normais do Java. Na verdade, os BPMObjects se comportam como JavaBeans, então esta limitação deve ser suprimida em alguma versão futura.

**Contorno** : O contorno atual é criar um BPMObject e utilizar herança para utilizar informações de classes já existentes.

**2 - Somente um BPMObject pode ser passado como parâmetro para uma JSP.** Este problema é até mais grave do que o anterior. Pela Figura Q2 podemos observar que somente um campo BPMObject pode ser selecionado. Isto nos força a criar um BPMO com a “aparência” da tela, ou ao menos com os campos que aparecem na tela. Uma coisa muito comum em um projeto Web é um pool de objetos sendo enviado entre as telas.

**Contorno** : O que fazemos usualmente é criar um BPMO por JSP, e por uma questão de facilidade na busca criamos a JSP com o mesmo nome do BPMO. Dentro do BPMO, colocamos campos e através de herança reutilizamos outros objetos.

Isto é apenas um contorno, pois os valores de uma instância de BPMO não pode ser compartilhados entre vários BPMOs (mas apenas sua estrutura). Somos obrigados a popular os valores cada vez que uma nova tela precisa ser apresentada. A Oracle deveria refatorar a tela da Figura Q2 para permitir a passagem de vários parâmetros para as JSPs, o que reduziria grandemente a quantidade de cópias dos valores que somos obrigados a fazer nos projetos atuais.

**Figura Q2.** Passagem de parâmetros para JSPs

Implementation Type  
BPM Object Interactive Call

Select BPM Object Variable: var1

Use BPM Object Presentation

Use JSP Presentation: jsp\_screen.jsp

Input  Display

**3 - Não existe um editor JSP dentro da ferramenta.** Existem vários editores JSP para o Eclipse disponíveis na internet. A Oracle poderia agregar um deles à ferramenta, tornando mais fácil a edição.

**Contorno** : Atualmente podemos baixar e instalar qualquer um, desde que a versão de Eclipse do ALBPM seja compatível com este plugin. Isto já nos traz um fator de produtividade maior para a ferramenta.



www.portalbpm.com.br

**4 - JSPs somente podem ser utilizadas a partir de Screenflows.** Este comportamento é estranho, já que Presentations podem ser utilizadas em Fluxos de processos ou Screenflows. Já páginas JSP somente podem ser utilizadas dentro de ScreenFlows.

As diferenças básicas entre um ScreenFlow e fluxos de processos são :

- Um ScreenFlow está atrelado a um usuário apenas, enquanto que um fluxo permite vários perfis entre as atividades. Na verdade, um ScreenFlow executa uma sequência de ações para um perfil específico.
- ScreenFlows não mantém estado da execução e instâncias de fluxo mantém estado ao longo do tempo. De uma forma bem simples : Se a energia acaba e a ferramenta é reiniciada, o ScreenFlow começa a execução a partir da primeira atividade definida. Já uma instância de fluxo de processo continua a execução de onde parou.

**Contorno** : O fabricante recomenda que interações com o usuário sejam feitos a partir de Screenflows. Esta é uma prática que deve ser seguida, pois no futuro podem ser o único mecanismo. O que não é explicado é como utilizar uma tela em ambientes transacionais, e que portanto precisam manter estado.

## Relativos à utilização de presentations

**5 – Editor de presentations muito limitado.** A versão 6.X da ferramenta incorpora um editor antigo e muito limitado. Ele trata a tela como um array de elementos, e para colocarmos um elemento a posição XY para onde arrastamos deve estar vazia. Na versão 10 da solução, já sob o controle da Oracle, o editor foi muito melhorado, tornando mais fácil a criação de uma presentation.

**Contorno** : Migrar para a versão 10 da ferramenta é a única forma de ter um editor melhorado para presentations, já que não existem outros editores na internet, já que o formato é proprietário.

**6 – Impossível atrelar códigos JavaScript a elementos de uma presentation.** A presentation é um tipo de tela com recursos limitados para interação com usuário. Não é possível a criação de componentes mais sofisticados através de AJAX ou JavaScript. Também não existe migração entre uma presentation e uma JSP. Ou seja, se foram utilizadas presentations e se encontrou algum limite, é necessário recriar os códigos via JSP, pois não há uma ferramenta para converter uma Presentation em JSP e vice-versa (seria talvez uma das mais úteis na ferramenta).

**Contorno** : Seria muito importante que um conversor de presentations para JSPs (e vice versa) fosse criado, pois agilizaria o processo de criação. As telas poderiam ser criadas como presentations e convertidas para JSP para adicionar dinamismo através de JavaScript. O ideal seria uma ferramenta “round-trip”, deixando o código ir de uma implementação para outra de forma facilitada. Um bom editor de JSP faz com que nos esqueçamos completamente das presentations.

**7 – Presentations presas dentro de um BPMObject.** A implementação de uma presentation é algo que foi feita de uma forma muito estranha, pois uma presentation é parte de um BPMObject. Podemos ter várias presentations dentro de um BPMObject, mas não podemos ter uma presentation formada por vários BPMObjects.

Isto é totalmente contra qualquer padrão Web, onde uma página é formada por composites, que são pequenos pedaços de código que fornecem funcionalidades. Não é possível, por exemplo, utilizar uma mesma presentation para mostrar dados de BPMObjects diferentes.

**Contorno** : Seria muito importante que a Oracle fizesse presentations e BPMObjects independentes. Na verdade, tenho a impressão de que um bom editor de JSP atrelado à ferramenta já resolveria este problema também, sepultando de vez as presentations.



www.portalbpm.com.br

**8 – Campos de tela não são relacionados com perfis.** É comum em atividades com interação humana que uma mesma tela seja apresentada a vários perfis, com ligeiras diferenças. Um campo pode ser apresentado mas não editado pela atendente, e apenas o perfil gerente teria direitos para alteração. No caso do ALBPM, uma presentation é totalmente desconectada do perfil que a executa. Algumas situações reais seriam mais simples de implementar se houvesse formas de correlacionar campos da tela com perfis específicos.

**Contorno :** O contorno atual é a criação de uma JSP. O BPMO passado como parâmetro teria o perfil que está executando como um atributo e um Scriptlet ou TAGLIB seria capaz de apresentar ou não determinada informação. Este contorno hoje é totalmente programático.

## BPMObjects

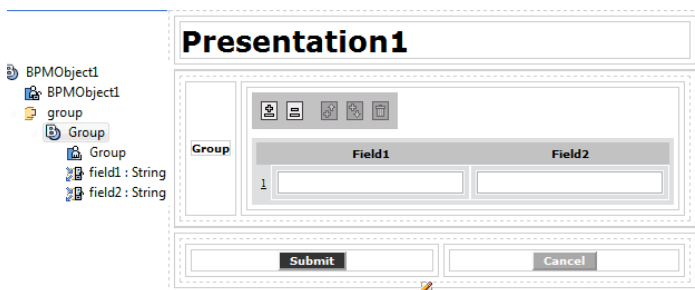
Os BPMObjects são como que os componentes de negócios da arquitetura do ALBPM. Eles carregam dados entre as atividades e seus campos podem ser utilizados em tomadas de decisão, por exemplo. Como são XMLs, podem ser versionados de forma a termos várias versões em execução em um dado momento. Quando são publicados, se tornam uma espécie de JavaBeans, que podem ser manipulados até via Java. Na teoria, é o ideal, pois desobriga o desenvolvedor a utilizar uma ferramenta Java para criar códigos e depois importá-los na ferramenta. Tornam o processo de criação dos objetos algo totalmente visual. Eles têm íntimo relacionamento com as telas, já que em uma presentation ou em uma JSP somente podem ser apresentados dados de um BPMObject.

### Utilização de BPMObjects

**9 – Somente um nível de hierarquia.** Quando desejamos criar um Objeto com uma coleção de outros objetos podemos utilizar um Group (Figura Q3) . O Group permite criar coleções de elementos, que visualmente se tornam muito fáceis de serem editados via presentation. Esta característica da ferramenta é interessante, mas se torna um limitante, já que temos apenas um nível para groups. Ou seja, não podemos ter um group dentro de outro group. Visualmente entendo que seria complexa a criação de uma tela deste tipo, mas em termos de objetos isto é uma necessidade.

**Contorno :** O contorno neste caso é a utilização da herança de objetos, para criarmos elementos dentro de elementos. Isto torna a estrutura do objeto mais próxima de um JavaBeans, mas torna o BPMObject mais distante de uma fácil implementação via presentation.

**Figura Q3.** Groups e sua identidade visual

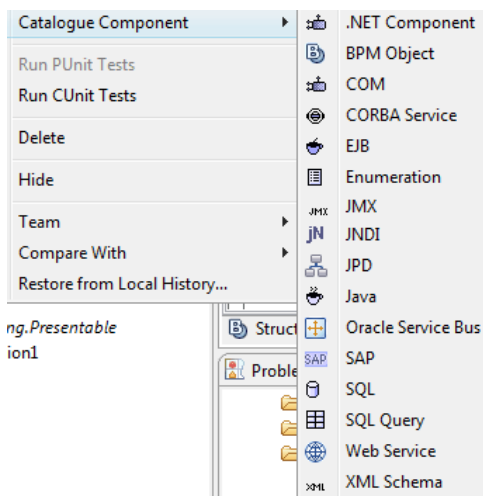




## Integração

Para permitir integração com legados o ALBPM incorporou um mecanismo para catalogar WebServices e uma diversidade de códigos no legado. O catálogo é feito de uma forma muito simples, apontando para o “endpoint” do WebService e diversos códigos são criados dentro da ferramenta. Internamente ele utiliza a API XMLBeans que cria classes derivadas de XMLObject para representar cada um dos objetos.

**Figura Q4.** Catalogo de componentes



## Utilização de Serviços integrados

**9 – Completo desacoplamento entre as telas e os serviços.** Imaginando uma sequência de passos para criação de um aplicativo, criamos a tela e os campos a serem editados. Uma vez acionada a tela, os valores dos campos são coletados e as informações serão passadas aos WebServices, para geração de novos dados para apresentação. O grande problema neste caso é que as telas somente entendem BPMObjects e os serviços somente entendem XMLObjects (XMLBeans). Enquanto que a criação de telas e catálogo dos serviços podem ser feitas de forma totalmente visual (e normalmente é o que se apresenta nas pré-vendas), a ligação dos dados coletados com a chamada dos WebServices deve ser feita com longos trechos de script, como mostrados a seguir :

```
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
  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
```

E acredite-me : Para cada tela a ser apresentada ou cada campo coletado que precisa ser armazenado scripts deste tipo precisam ser criados. O trabalho pode ser exaustivo, e um projeto que exige integração irá fazer a produtividade da ferramenta cair por terra abaixo. Algo que poderia ser rápido

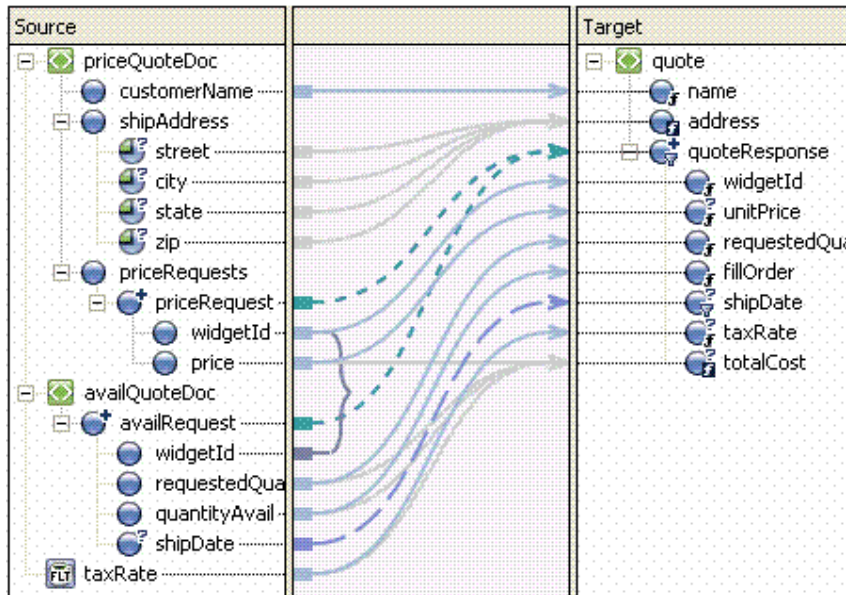


www.portalbpm.com.br

e simples se torna demorado e totalmente sujeito a erros, pois uma atribuição errada pode tornar todo o sistema instável.

**Contorno** : A sugestão seria a Oracle incorporar o Mapper existente no WLI dentro do ALBPM. Ao invés de longos trechos com scripts os mapeamentos entre BPMObjects e XMLObjects poderiam ser feitos de forma visual. Isto tornaria a ferramenta mais produtiva e menos sujeita a erros de edição nos scripts.

**Figura Q5.** Mapper incorporado ao ALBPM



## Desenvolvimento X Produção

A ferramenta de desenvolvimento é o Oracle BPM Studio. Até a versão atual é um Eclipse com vários plugins que permitem a edição dos vários tipos de elementos gráficos. Incorporado à solução existe um TOMCAT que roda uma mini instância do aplicativo. Quando o programa é colocado em produção, uma nova versão “enterprise” é disponibilizada.

**10 – Versão de desenvolvimento não é idêntica à de produção.** Como foi criada uma simplificação da versão de produção, uma série de recursos não estão disponíveis na versão de desenvolvimento. O usuário não se loga no workspace, por exemplo. Somente a versão de produção permite utilizar um LDAP ou base de dados para validar usuários. Um aplicativo mais avançado provavelmente irá precisar acessar informações do LDAP e tomar decisão dentro do fluxo de processos. Este tipo de implementação não pode ser simulado no workspace. A maioria das implementações que utilizam a PAPI não são facilmente simuladas dentro do workspace. Mesmo que possam ser simuladas, a ferramenta de administração e log viewer do ambiente de produção não estão disponíveis dentro do Workspace.

**Contorno** : A sugestão seria utilizar a mesma versão de produção dentro do studio, atachando um servidor REAL como um processo a ser depurado. Desta forma teríamos a certeza de que toda a implementação funcionaria da mesma forma em ambiente de produção ou durante o desenvolvimento. A IBM há muito tempo atrás tentou implementar servidores leves para



www.portalbpm.com.br

desenvolvimento (na época do Visual Age Java, que tinha um mini servidor), mas acabou abortando a idéia e hoje as ferramentas de desenvolvimento IBM executam um servidor Websphere comum, a custo de mais memória da máquina sendo consumida. Imagino que este deveria ser o caminho a ser seguido pela Oracle. Na verdade, isto pode ser feito hoje, com um pouco de trabalho. O servidor de produção pode ser iniciado em modo debug (opção -Xrunjdw) e atachado ao eclipse como um processo externo.

**11 – Falta de um depurador visual.** Atualmente o mecanismo para depuração no ALBPM é aquele mesmo utilizado na pré historia, que é adicionar LOGs à execução do processo e acompanhar as mensagens no console. Não existem mecanismos como break points para que paremos em alguma atividade, ou analisemos como estão as variáveis em dado momento da execução. Depurar um programa em ALBPM exige uma arte que mistura intuição e observação do código sendo executado. A maioria das exceptions são capturadas pela ferramenta e são lançadas outras de mais alto nível, omitindo muitas vezes informações importantes sobre o erro

**Contorno :** Bem, creio que a Oracle têm um bom trabalho a ser feito aqui. Da mesma forma que ela conseguiu criar depuradores sobre o ALSB na nova versão, que é praticamente todo composto por XMLs, creio que o mesmo possa ser feito sobre o XPDL atualmente existente no ALBPM. Enquanto isto não acontece, ficamos presos ao mecanismo de mensagens no console.

## Conclusões

Embora neste texto apresentemos pontos a serem melhorados, o ALBPM é uma das soluções que mantém um bom equilíbrio entre elementos visuais e quantidade de códigos criados. Ela fica no meio termo entre uma ferramenta totalmente programática, onde tudo pode ser feito mas na forma de códigos, e uma solução totalmente visual, onde nenhum código é permitido e a solução têm seus limites bem definidos. Creio que agora, sob nova direção, ela possa receber a injeção necessária para se tornar uma das soluções mais avançadas do mercado. O mercado demanda cada vez mais por soluções completas que tragam formas simples de implementação. Isto poderá fazer com que os longos ciclos de desenvolvimento sejam cada vez menores, tornando viáveis implementações sob as soluções BPMS.

## Links

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

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

Site com as documentações das versões e download das ferramentas