Desenvolvimento de um Sistema de Protocolo Web Baseado em Padrões de Projeto

Sérgio Antônio Martini Bortolin Júnior, Pedro Conrad Júnior, Émerson de Oliveira Rizzatti,
Diego Luís Kreutz, Robson Romário Gonçalves
Núcleo de Tecnologia da Informação e da Comunicação - NTIC
Grupo de Pesquisa em Sistemas de Informação - GPSI
Universidade Federal do Pampa - UNIPAMPA
Av. Tiarajú, 810 - Alegrete, RS - 97546-550 - Brasil

{sergiojunior, pedrojunior, emersonrizzatti, diegokreutz, robsongoncalves}@unipampa.edu.br

RESUMO Este artigo apresenta o Sistema de Gestão de Protocolo (SGP), uma solução de *workflow* desenvolvida pela Universidade Federal do Pampa (UNIPAMPA), para gerenciar de forma dinâmica qualquer processo de negócio organizacional. O sistema foi concebido para utilização em órgãos públicos, com o intuito de ser flexível, permitindo a configuração do sistema utilizando opções de parametrização e integração com os demais sistemas institucionais que dependam de tramitações de processos. Para prover tal interoperabilidade foi implementada uma solução baseada em *plugins* customizáveis, programados utilizando um modelo de regras específicas que são acionadas em determinados estados de um fluxo de processo, sendo este o principal diferencial do sistema em relação aos demais aplicativos do gênero.

1. INTRODUÇÃO

O Sistema de Gestão de Protocolo (SGP) foi planejado, projetado e desenvolvido pela Universidade Federal do Pampa (UNIPAMPA), objetivando controlar via *web* os processos de negócio institucionais, e ao mesmo tempo comunicar-se com os demais sistemas da universidade que dependem de tramitações de processos. O SGP é baseado no conceito de *workflow,* que, segundo Araújo & Borges (2001), tem como objetivo "oferecer soluções sobre como gerar, armazenar, compartilhar e rotear documentos em organizações, visando a diminuição da manipulação física de documentos em papel".

Sistemas de *workflow* encaixam-se na idéia de gestão baseada em processos e BPM (*Business Process Management*), pois através de tramitações permitem representar cada estado do processo, seus participantes e documentos associados. No contexto da UNIPAMPA, além da diminuição dos gastos com papel e impressão, o desenvolvimento de um sistema de *workflow* foi principalmente motivado pela automatização dos processos de negócio institucionais, os quais são mapeados utilizando-se a metodologia BPM.

Durante o planejamento da arquitetura do sistema levou-se em conta que uma das principais

características deveria ser a componentização, possibilitando que as demandas futuras de customizações para os processos não tenham impacto na estrutura principal do sistema. Além disso, também foi levado em conta que diversos sistemas institucionais podem comunicar-se com o protocolo, fato este que direcionou o desenvolvimento do sistema de forma a tornar-se uma solução de *workflow* extensível. Assim, adotou-se uma solução baseada em *plugins*, unidades de código independentes, que podem ser ativadas e desativadas dinamicamente na configuração do fluxo de processo referente ao sistema externo integrado.

2. TRABALHOS RELACIONADOS

Inicialmente, realizou-se um estudo comparativo entre algumas soluções de protocolo atualmente disponíveis no mercado, baseando-se numa avaliação de recursos, conforme mostra a tabela 1.

Legenda: Integr = Integração com Sistema Externo, N/D = Não divulgado

Sistema	Licença	Plataforma	Plugins	Padrões projeto	Integr.	Prazos	Site
ERP AIX	Paga	Windows	N/D	N/D	N/D	Sim	http://www.aix.com.br
Protocolo Web	Paga	Web	Não	N/D	Não	Não	http://www.mplopes.com.br/
CuteFlow	Gratuita	Web	Sim	N/D	Não	Sim	http://www.cuteflow.org/
ProcessMaker	Gratuita	Web	Sim	N/D	Sim	Sim	http://www.processmaker.com/
Galaxia	Gratuita	Web	Sim	N/D	Não	Sim	http://workflow.tikiwiki.org/tiki-index.php
Sol	Paga	Web	Não	N/D	Não	Sim	http://www.pixsoft.com.br/prod_sol.htm
Maestro	Paga	Web	Não	N/D	Sim	Sim	http://www.maestro.inf.br/

Tabela 1 - Comparativo de Sistemas de Protocolo

Foram considerados como atributos principais no comparativo os seguintes recursos: plugins, padrões de projeto, integração com sistemas externos e controle de prazos de tramitações. Obtiveram-se tais dados analisando o site da ferramenta e a documentação, quando disponível.

Constatou-se que a maioria dos sistemas não prioriza ou enfatiza a integração com sistemas externos, os quais dependem de tramitações de processos. Também verificou-se que com relação a metodologias e padrões de projeto não existe uma divulgação por parte das empresas/ comunidades desenvolvedoras.

Como justificativa por parte da UNIPAMPA de desenlvolver seu próprio sistema de protocolo, pode-se considerar a facilidade de integração com os demais sistemas institucionais (os quais dependem de tramitações de processos), com base em plugins e parâmetros de integração. Também pode-se levar em conta a possibilidade de integração com as ferramentas de modelagem de processo de negócio utilizadas pela instituição, automatizando a parte cadastral do processo de negócio.

3. METODOLOGIA E FERRAMENTAS

Utilizou-se como metodologia de desenvolvimento o modelo interativo e incremental, conforme destaca Bezerra (2007): “Em cada ciclo de desenvolvimento podem ser identificadas as fases de análise, projeto, implementação e testes. Esta característica contrasta com a abordagem clássica, na qual as fases de análise, projeto, implementação e testes são realizadas uma única vez”.

A etapa de análise caracterizou-se por reuniões entre os desenvolvedores e os futuros usuários do sistema, com demonstração de protótipos, troca de ideias e sugestão de melhorias.

Na fase de projeto definiram-se a arquitetura do sistema, os padrões de projeto a serem adotados e a forma como o SGP se comunicaria com sistemas externos. Desenvolveu-se também os diagramas UML de arquitetura e o Documento de Arquitetura de Software (DAS), baseando-se no artefato de DAS da metodologia RUP (*Rational Unified Process*).

A fase de implementação baseou-se numa abordagem mista entre a programação estruturada e a programação orientada a objetos, utilizando-se de recursos dos dois paradigmas. Foram adotadas também normas de nomenclatura e notações definidos pelo time de desenvolvimento.

A etapa de testes esteve fundamentada basicamente em testes de interface, considerando o modelo clássico “caixa-preta”. Segundo Pressman (1995), “este teste possibilita que o engenheiro de software derive conjuntos de condições de entrada que exercitem completamente todos os requisitos funcionais para um programa”.

Para a execução do SGP foram utilizadas principalmente as seguintes tecnologias: a) a linguagem de programação PHP (Melo & Nascimento, 2007), versão 5.3; b) o sistema gerenciador de banco de dados PostgreSQL, versão 8.4; c) o servidor Web Apache, versão 2.0. Além destas, também foram utilizados dois frameworks de desenvolvimento, a saber: a) o framework PHP CodeIgniter, versão 1.7; b) o framework JavaScript jQuery (Silva, 2010), versão 1.4.

Para a codificação do sistema foram utilizadas as seguintes ferramentas: a) o ambiente de desenvolvimento integrado (IDE) Netbeans 6.9; b) o software pgAdmin, versão 1.10, para a manipulação do banco de dados; c) o repositório de códigos Subversion (SVN); d) a ferramenta CASE Enterprise Architect, versão 7.5; e) o sistema de gestão de projetos Redmine, versão 1.1.

Com relação à tecnologia, *frameworks* e ferramentas, a prioridade foi dada a utilização de softwares gratuitos, sendo que a maioria das soluções para desenvolvimento adotadas, citadas acima, é distribuída sob licença *open-source*.

4. ARQUITETURA DO SISTEMA

4.1 Visão Geral

A arquitetura do SGP foi desenvolvida utilizando-se o modelo "Web 3 Camadas", ilustrado na figura 1.

Figura 1 - Aplicação Web 3 Camadas

A arquitetura Web 3 Camadas apresenta algumas vantagens sobre a arquitetura cliente-servidor de 2 camadas, conforme observado por Ramirez (2000):

- É mais fácil de modificar ou substituir uma camada sem afetar as demais;
- Separar as funcionalidades de aplicação e banco de dados resulta em melhor balanceamento de carga;
- Políticas de segurança podem ser aplicadas nas camadas de servidor sem prejuízo às camadas clientes.

Como as regras de negócios ficam alocadas apenas no lado servidor, no modelo de 3 camadas as atualizações de versões são centralizadas num único ponto, não sendo necessário atualizar o sistema no lado cliente, o que é algo importante para a sustentabilidade da solução. Uma outra vantagem, observada por Engholm (2010), é que "pode-se utilizar browsers gratuitos de mercado, e a instalação dos mesmo nas máquinas-clientes é extremamente simplificada".

No SGP a camada de lógica de negócio também é dividida em camadas, utilizando-se o *design pattern* MVC (Models, Views e Controllers), formando assim uma sub-arquitetura do modelo ilustrado na figura 1. O padrão MVC divide a aplicação em camadas distintas, conforme observa Melo & Nascimento (2007):

- A camada Modelo (Model) trata da parte de comunicação com banco, fluxo de dados e lógica de negócio.
- A camada de Visualização (View) formata os dados para visualização.
- A camada de Controle (Control) direciona o fluxo da aplicação fazendo a ponte entre a Visualização e o Modelo.

4.2 Padrões de Projetos Utilizados

Ao curso do desenvolvimento do SGP foram utilizados os padrões de projetos fornecidos pelo framework adotado, eliminando a necessidade de "reinventar a roda". Dentre os padrões, pode-se citar: MVC, *Front Controller, Active Record, Singleton, Composite View e Observer*.

O padrão MVC, conforme citado na seção anterior, divide a aplicação em três camadas: Models (Modelos), Views (Visões) e Controllers (Controladores). O propósito desta separação é a manutenibilidade, facilitando a alteração no código-fonte, tanto para fins de correção quanto para inclusão de novas funcionalidades.

O *Front Controller* é um padrão que propõe a utilização de uma classe para receber todas as requisições que chegam ao sistema. Nesta classe podemos anexar um comportamento comum a todas requisições, como por exemplo, enviar um comando ao navegador para não armazenar em cache a página que o servidor devolverá.

O *Active Record* é uma forma de representar uma tabela do banco de dados como classe, e seus registros como objetos, onde em tal classe residiram os métodos para manipular tal tabela. Além de conter métodos para manipular a tabela, pode conter também métodos relacionados a lógica de negócios da aplicação, conforme observa Dall'Oglio (2009): "Com o pattern Active Record, temos uma única camada, na qual temos lógica de negócios (modelo conceitual) e métodos de persistência do objeto na base de dados (gateway)".

O padrão *Singleton* propõe uma classe que deve possuir apenas uma instância, sendo que tal instância deve ser acessada globalmente. Tal padrão é utilizado no SGP, por exemplo, para acessar os parâmetros de configuração da aplicação, classe esta que não necessita de mais de um objeto.

O padrão *Composite View*, conforme observado por Rocha (2003), tem como objetivo "criar um componente de View a partir de Views menores para dividir as responsabilidades, simplificar a construção da interface e permitir o reuso de componentes da View". Cada objeto de *View* do SGP é gerado partindo deste conceito, através de componentes menores e comuns.

A utilização do padrão *Observer* é explicada na seção seguinte.

4.3 Suporte a Plugins

Determinados fluxos de processos, quando atingem um determinado estado na sua movimentação, devem comunicar determinadas partes do sistema sobre tal mudança de estado. Com base nisso, desenvolveu-se uma solução baseada no padrão de projeto *Observer*, que segundo Melo e Nascimento (2007), este padrão "tem como proposta definir uma dependência de um para muitos entre os objetos que, quando um objeto muda de estado, todos os objetos dependentes são notificados e atualizados automaticamente, se necessário".

O padrão sustenta a existência de pelo menos dois tipos de objeto, denominados: *Subject* (Observado) e *Observer* (Observadores). Conforme destaca Melo e Nascimento (2007), "Um Subject conhece seus observadores, sendo que este pode ter vários observadores, esse também

fornece um meio de acrescentar e remover objetos do tipo *Observer*. Já o *Observer* define uma maneira de atualização para objetos que deveriam ser notificados sobre mudanças em um Subject (Observado)". A figura 2 ilustra o relacionamento entre as classes do SGP.

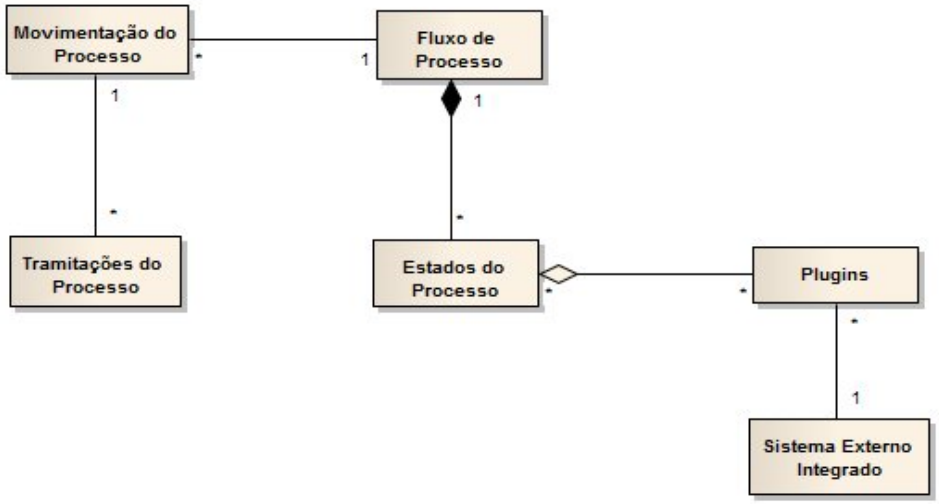

Figura 2 - Diagrama UML de Classes SGP

Com base na figura 2, no SGP o objeto *Subject* é representado pela entidade "Movimentação do Processo", e o(s) objeto(s) observador(es) são representados pela entidade "Plugins", acionados sempre quando o *Subject* mudar de estado. O conhecimento dos objetos a serem notificados fica alocado nos plugins (*Observer*) que estão associados ao *Subject,* ou seja, o *Subject* não precisa saber quais objetos serão notificados, mas seus observadores precisam, resultando assim numa baixa acoplagem do Subject aos objetos que serão notificados.

Como exemplo prático, consideramos que possuímos um sistema externo integrado ao SGP, para automação de pedidos de compra. Assim, no SGP, quando o fluxo de processo referente ao pedido de compra atingir um estado chamado "Aprovar Pedido", serão acionados os plugins vinculados a tal estado, os quais encapsulam a regra de negócio que definirá o status do pedido como "Aprovado". Assim, quem notificará objeto de pedido será o *Observer* (Plugins) e não o *Subject* (Movimentação do Processo).

Desta forma, a solução baseada no padrão de projeto *Observer* forneceu uma forma elegante de resolver o problema de integração entre sistemas, não "poluindo" o código-fonte núcleo do SGP com regras de negócios de sistemas externos.

5. RESULTADOS E PERSPECTIVAS

5.1 Definição de Papéis

Observando as necessidades institucionais quanto à tramitação de documentos, foi constatada a necessidade de centrar o trabalho em grupos de pessoas e não em pessoas específicas, o que pode ser embasado conforme Araújo e Borges:

"A rotatividade de pessoas dentro de uma organização é constante e estas mudanças deveriam ser constantemente atualizadas nos fluxos de trabalho em execução. Em geral, cargos ou papéis costumam permanecer estáveis por mais tempo dentro de uma organização. Pensando assim, definir papéis, ou seja, um conjunto de características ou responsabilidades necessárias para a execução de alguma tarefa dentro do processo e associar tais papeis à execução das atividades torna as definições de fluxos de trabalho mais flexíveis" (Araújo & Borges, 2001).

Em face disso, a definição de papéis é de suma importância para o bom andamento dos processos organizacionais, objetivando que a questão da rotatividade influencie o mínimo possível sobre o sistema.

5.2 Extensibilidade

O SGP atua lado a lado com a modelagem de processo de negócio para uniformizar a tramitação de processos, focando na integração e flexibilidade como pontos principais de seu desenvolvimento. Estas características tornam-se evidentes pela extensibilidade adotada no modelo de desenvolvimento, baseado em plugins. Na UNIPAMPA, o SGP foi integrado ao sistema de pedidos de compra (atualmente em fase de testes) para controlar o campo de status do pedido com base nas tramitações do processo, mostrando resultados satisfatórios até o momento.

5.3 Uso da Modelagem BPM

A grande característica presente na utilização de modelagem BPM é a de que se permite o alinhamento dos processos institucionais visando à qualidade do serviço prestado pela instituição, assim como monitorar os ciclos evolutivos dos processos, buscando sempre a maturação destes processos. O SGP fornece um ambiente que proporciona o armazenamento e a execução dos processos modelados, servindo, assim, como interface de *workflow*.

5.4 Perspectivas

Como perspectivas de melhorias futuras do sistema podemos enumerar as seguintes: a) novos nívies de integração com ferramentas de modelagem BPM; b) acompanhamento visual de tramitações processo a processo; c) integração com sistemas desenvolvidos futuramente através de novos plugins; d) criação de um pacote de distribuição da aplicação para outras instituições.

6. CONCLUSÃO

Este artigo apresentou o Sistema de Gestão de Protocolo (SGP) desenvolvido pela UNIPAMPA. Ao longo do desenvolvimento do SGP foram utilizados padrões de projeto propostos pela engenharia de software, os quais se encaixavam na resolução dos problemas encontrados, proporcionando soluções de qualidade que visam à reutilização de código e à manutenibilidade do sistema.

No contexto da instituição, estima-se que a adoção do SGP para automatizar os processos de negócio instituicionais resultará em benefícios significativos, como a agilidade de execução destes processos e a diminuição dos custos com papel e impressão. Ao mesmo tempo, fornecerá um repositório para centralizar todos os processos institucionais mapeados e proverá a interoperabilidade necessária com os demais sistemas da instituição.

No atual estágio do SGP, constata-se também que a solução de integração com sistemas externos, baseada em plugins, é uma característica que agrega flexibilidade ao sistema, tendo em vista que os demais órgão públicos que vierem a utilizar o SGP no futuro, também poderão criar seus plugins personalizados, sem necessitar entender de aspectos da arquitetura ou alterar o código-fonte principal do SGP, adotando, desta forma, uma solução de *workflow* extensível.

7. REFERÊNCIAS

Araújo & Borges (2001) - Disponível em: http://equipe.nce.ufrj.br/mborges/publicacoes/Apostila%20JAI2001.pdf, acesso em 10/03/2011.

Bezerra, E. **Princípios de Análise e Projeto de Sistemas com UML, 2 Edição**. Campus, Rio de Janeiro-RJ, 2007.

Dall'Oglio, P. **PHP - Programando com Orientação a Objetos, 2 Ediçao.** Novatec, São Paulo-SP, 2010.

Engholm, H. **Engenharia de Software**. Novatec, São Paulo-SP, 2010.

Melo & Nascimento. **PHP Profissional**. Novatec, São Paulo-SP, 2007.

Minetto, E.L. **Frameworks para Desenvolvimento em PHP.** Novatec, São Paulo-SP, 2007.

Pressman R. **Engenharia de Software**. Makron Books, São Paulo-SP, 1995.

Ramirez (2000) - Disponível em: http://www.linuxjournal.com/article/3508, acessado em 10/03/2011.

Rocha (2003) - Disponível em http://www.argonavis.com.br/cursos/java/j550/j550_13.pdf, acessado em 10/03/2011.

Silva, M. S. **jQuery - A Biblioteca do Programador JavaScript**, **2 Edição**. Novatec, São Paulo-SP, 2010.