Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Trabalho de Formatura Supervisionado
Bacharelado em Ciência da Computação
Monografia de Estágio Supervisionado

Business Process Management e Workflow - jBPM

Filipe Ferraz Salgado

Supervisor: Francisco Reverbel

São Paulo, dezembro de 2006

Sumário

I. Parte Técnica
1. Introdução.....03
   1.1 O que é um Workflow Management System (WFMS).....04
   1.2 O argumento para o workflow.....05
2. Conceitos e Tecnologias estudadas.....07
   2.1 Um olhar mais aprofundado.....07
   2.2 As implementações de workflow.....11
   2.3 A utilização do jBPM.....13
   2.4 Outras ferramentas utilizadas.....14
3. Atividades realizadas.....15
4. Resultados obtidos.....18
5. Conclusão.....19
6. Bibliografia.....20

II. Parte Subjetiva
1. Desafios e frustrações.....22
2. Matérias mais importantes do BCC no trabalho.....23
3. Interação com o responsável e com a empresa.....25
4. Aprimoração.....26

Parte I – Parte Técnica

Capítulo 1 - Introdução

Hoje em dia, as empresas estão cada vez mais interessadas em realizar suas tarefas de algum modo eletrônico, de forma a minimizar os gastos e maximizar a produtividade, por exemplo, compras on-line, e-mails. Outra tecnologia que vem sendo adotada é o controle do fluxo de trabalho (workflow) através de um gerenciador de tarefas. De uma maneira geral, o gerenciador permite que se designem tarefas a vários funcionários da empresa e que, através do software, eles realizem suas tarefas. Além disso, é possível saber qual funcionário falta realizar sua tarefa. Isso tudo facilita o controle de um processo dentro de uma empresa. O sistema ConteXpress, software que faz gestão eletrônica de documentos, da Murah Technologies já possuía um gerenciador de tarefas e o objetivo do trabalho inicialmente era verificar a viabilidade de uma atualização da versão utilizada para uma versão mais recente, gerando um arquivo compactado com o código fonte da nova versão no sistema, mas devido à necessidade o trabalho se estendeu e chegamos a implantar a nova versão.

Começarei essa introdução a workflow discutindo a sua relação com business process management (BPM). O termo workflow é usado para denotar uma série de interações entre as pessoas e um sistema de computador, é também mais frequentemente utilizado entre desenvolvedores. Às vezes, o seu significado está sendo restrito às diferentes telas da interface do usuário. BPM tem um escopo mais abrangente do que workflow, já que esse é usado em um contexto mais técnico. Ele implica também os temas não-técnicos como análise, impacto organizacional, do ponto de vista de um gerente.

Problemas relacionados a workflow são aqueles nos quais temos uma série de estados a serem executados a partir de um inicial. Dessa forma, podemos pensar em uma máquina de estados finitos, onde cada estado pode representar uma ação humana (autorização ou reprovação), um estado de espera ou ainda uma tarefa a ser executada.

Um exemplo simples seria um pedido de férias por parte de um funcionário. Ele iniciaria o processo e o seu chefe faria uma avaliação, autorizando, ou não, a sua dispensa. Caso reprovasse, o processo terminaria ali, mas se autorizasse, o processo poderia ir para o seu superior para que ele desse uma autorização final.

O que é um Workflow Management System (WFMS)

Definições

Um workflow management system (WFMS) é um componente de software que recebe como entrada uma descrição formal de processos e mantém o estado das execuções dos processos, podendo assim delegar atividades para pessoas e aplicações.

Para começar, temos que definir alguns conceitos básicos: uma definição de processo e uma instância de processo. Uma definição de processo é uma descrição formal de um processo. Uma instância de processo é uma execução de uma definição de processo. Mas o modo como expressamos os passos de um processo ainda não está muito bem definido. Esta é a maior diferença entre as especificações e ferramentas para workflow e BPM.

Outra importante responsabilidade de um WFMS é manter o contexto para cada execução de processo. Uma variável de contexto – ou simplesmente variável - é uma variável associada a uma instância de processo. Geralmente a definição de processo declara as variáveis de contexto. Então, no momento da criação da instância de processo, as variáveis são inicializadas.

Casos de uso

Uma opção é usar um WFMS como uma plataforma de enterprise application integration (EAI). Atualmente, na maioria dos ambientes empresariais, a infra-estrutura de TI tem várias aplicações diferentes e base de dados rodando em uma intranet. Tipicamente, essas aplicações têm um claro propósito quando elas são introduzidas em uma organização. Alguns exemplos são gerenciamento de clientes, entrada de pedidos, emissão de contas,... Vamos chamá-los de aplicações dedicadas. Cada uma dessas aplicações dedicadas contém conhecimento sobre o que elas têm que fazer. Como regra, poderia se dizer para usar uma aplicação dedicada se ela satisfizer suas necessidades. Usando um WFMS, você amarraria todas essas aplicações dedicadas e criaria uma plataforma EAI com ele.

Uma segunda opção onde WFMS tem grande utilidade é no desenvolvimento de software de workflow que possui várias tarefas relacionadas às pessoas. Para isso, a maioria dos WFMS's tem um mecanismo bastante consistente de criação de tarefas. Este caso de uso é especialmente produtivo para organizações que procuram certificados ISO ou CMM. Ao invés de documentar os procedimentos em formato texto, um WFMS permite que você crie processos automatizados dos procedimentos que são modelados como definições de processos.

A terceira opção é acrescentar uma ferramenta de workflow em outra aplicação. Lembre-se da primeira opção que uma aplicação dedicada tem uma série de atividades definidas. As companhias que desenvolvem as aplicações dedicadas podem adicionar o workflow nos seus softwares. A ferramenta de workflow é, nesse caso, usada apenas como um componente da aplicação, abstraindo-se dos usuários. A principal razão pela qual se deve acrescentar uma ferramenta de workflow em uma aplicação é a reutilização e manutenção do software.

O argumento para o workflow

Introduzir workflow em uma empresa traz benefícios tanto no nível de desenvolvimento de software quanto no nível comercial.

Desenvolvimento fácil

Um workflow management system irá facilitar o desenvolvimento e mais ainda a manutenção do software da empresa.

- **Risco de desenvolvimento reduzido** – o analista de negócios irá falar a mesma língua que o desenvolvedor (em termos de estados e ações). Isso quer dizer que o desenvolvedor não precisará fazer uma tradução dos requerimentos para o design de software.

- **Implementação centralizada** – é o processo que muda então a maior vantagem em usar um sistema de workflow é que a implementação não é uma combinação confusa de peças do software espalhadas em vários sistemas.

- **Desenvolvimento rápido da aplicação** – seu software é liberado da função de saber quem participa de um processo para ter assim um desenvolvimento mais rápido e um código que mais fácil ter ser mantido.

Business process management (BPM)

Antes que você possa automatizar seus processos, a mais difícil tarefa, mas também a mais recompensadora, é analisá-los e criar uma descrição formal. Um exemplo de site que fez uma boa descrição dos benefícios ganhados com essa análise é o e-workflow.org.

- **Maior eficiência** – automatização de vários processos resulta na eliminação de vários passos desnecessários.

- **Melhor controle do processo** – melhor gerenciamento de processos através da padronização dos métodos de trabalho e da disponibilidade das trilhas de auditorias (logs de execução).

- **Melhor serviço ao cliente** – consistência no processo leva a maiores níveis de eficácia na resposta para clientes.

- **Flexibilidade** – controle do software sobre o processo possibilita uma reestruturação conforme a necessidade.

Capítulo 2 – Conceitos e tecnologias estudadas

Um olhar mais aprofundado

Interfaces WFMS

Um WFMS recebe definições de processos como entrada. Por enquanto, pense que uma definição de processo é um diagrama UML de atividades, um diagrama UML de estados ou uma máquina finita de estados. Alguns exemplos são pedido de férias, envio de contas,... Então, o WFMS mantém o estado e o contexto das execuções dessas definições de processos. Para isso, ele precisa ser informado das mudanças de estado. As mudanças de estado das execuções de processo podem ser logadas para efeitos de monitoramento.

- **Definição:** A interface de definição de um WFMS permite que os desenvolvedores de processo coloquem em uso as definições de processos. Note na figura 1 que o ator ‘process developer’ ocupa a posição de uma combinação de um analista de negócios e um desenvolvedor de software.

- **Execução:** A interface de execução permite aos usuários e sistemas agirem nas instâncias de processos. Instâncias de processos são execuções de definição de processos. O controle de fluxo de uma definição de processo é a descrição de uma máquina de estados. Os dois métodos principais na interface de execução são: começar uma instância de processo e sinalizar o fim de um estado para o WFMS.

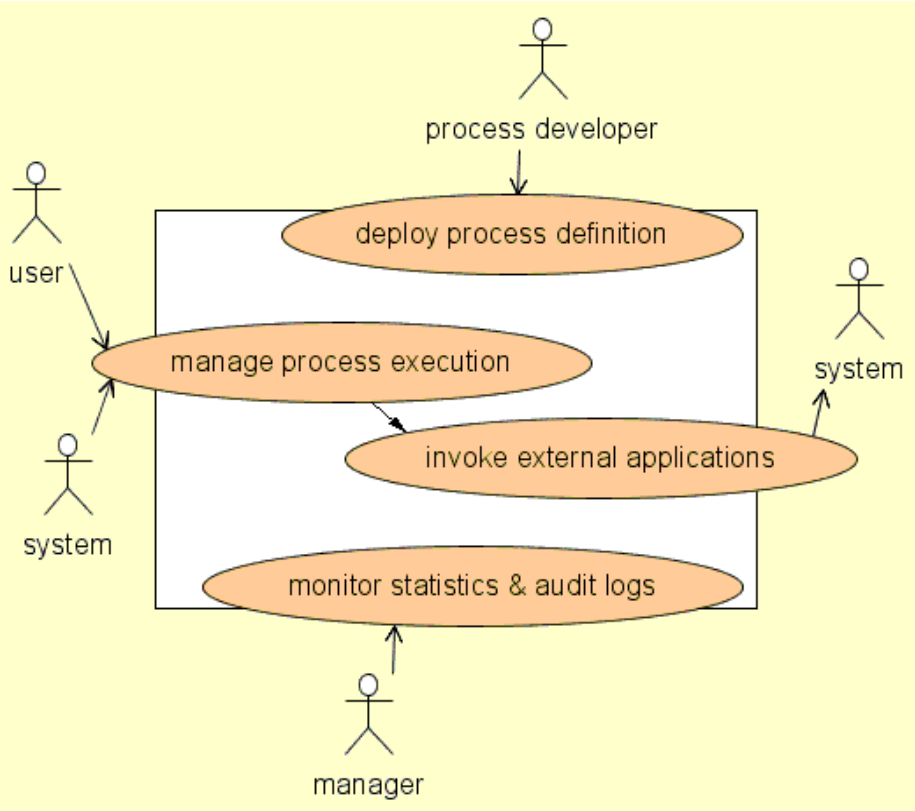

Figura 1: Interfaces de um WFMS

- **Aplicação:** A interface de aplicação denota a interação iniciada entre um WFMS e sistemas externos. Quando um usuário ou sistema gerenciam a execução de uma instância de processo, eventos são gerados. Uma definição de processo pode especificar que um código seja executado em um evento. O código pode comunicar com outros sistemas dentro ou fora da organização.

- **Monitoração:** Com a interface de monitoração, os gerentes podem ser auxiliados através das estatísticas dos logs de execução dos processos. Algumas vezes, os logs de execução também são chamados de trilhas de auditoria.

Essas são 4 das 5 interfaces definidas no modelo de referência da WfMC.

As 4 camadas de uma definição de processo

Nessa parte, eu tento responder a pergunta “Qual o conteúdo de uma definição de processo?” Esta síntese é baseada em princípios e conceitos que são usados nos modelos de várias especificações e

ferramentas de acordo com o artigo The State of Workflow by Tom Baeyens. Ela reflete o que tem em comum entre a maioria desses modelos. O conteúdo de uma definição de processo pode ser dividido em 4 camadas: estado, contexto, lógica de programação e interface de usuário.

A camada do estado

A camada de estado de um processo é simplesmente especificar os estados e o fluxo de controle. Fluxo de controle em linguagens de programação padrões é derivado das arquiteturas de Von Neuman, onde o fluxo de controle é determinado pela ordem que nós escrevemos as instruções, if-statements, loop-statements, e assim por diante. O fluxo de controle em um processo é basicamente o mesmo. Mas ao invés de usar uma instrução como elemento básico, o elemento básico em um processo é o estado.

Um estado (também conhecido como wait-state) em um processo especifica a dependência em relação a um ator externo. O sentido de um estado é como “Agora o sistema X ou pessoa Y tem que fazer alguma coisa. Espere aqui até que um sinal externo daquele ator sinalize que a tarefa foi feita.” Um exemplo típico de estado é um passo de aprovação.

Um estado em uma definição de processo também especifica de qual ator a execução depende. Os nomes dados a esses atores correspondem a noção de swimlanes em um diagrama de atividades. Com essa informação, o WFMS pode extrair as listas de tarefas (tasklists), que é a feature mais comum entre os WFMS’s. Como foi dito anteriormente, atores podem ser pessoa ou sistemas. Para os estados que requerem atores humanos, a pessoa tem que ser calculada em tempo de execução pelo WFMS. A dependência do WFMS em relação aos dados da organização origina-se desse fato.

O fluxo de controle de uma definição de processo é um conjunto de estados combinados com as relações entre os estados. A lógica entre os estados especifica quais caminhos de execução rodam concorrentemente e quais rodam de maneira exclusiva. Caminhos concorrentes são modelados com forks e joins, enquanto que caminhos exclusivos são modelados com decisions e merges. Observe que na maioria dos modelos, todos os estados têm um merge implícito.

Durante uma execução de processo, um token é um ponteiro usado pelo WFMS para manter o caminho do estado atual do processo. Isso corresponde à noção de contador do programa nas arquiteturas de Von Neuman. Quando um token chega a um estado, o token é atribuído ao agente externo pelo qual o WFMS está esperando. O agente externo pode ser um usuário, um grupo, ou um sistema de computador. Vamos definir um ator como sendo um usuário ou um sistema que atua em uma execução de processo. A atribuição de um ator a um token é a única ocasião onde o WFMS precisa acessar os dados da organização. Essa atribuição permite ao WFMS extrair as listas de tarefas (tasklists).

A camada do contexto

Uma variável de contexto do processo – ou simplesmente variável – é uma variável associada com uma instância de processo. Variáveis de processo permitem que um desenvolvedor de processo armazene dados durante o ciclo de vida da instância do processo. Alguns WFMS's têm um conjunto fixo de tipos, enquanto que em outros sistemas, é possível definir seus próprios tipos.

Observe que variáveis também podem ser usadas para armazenar referências. Uma variável poderia referenciar, por exemplo, um registro no banco de dados ou um arquivo no disco. Quando se deve usar referência depende de como outras aplicações usam os dados referenciados.

A camada da lógica de programação

Lembre-se que uma ação é um trecho de código a ser executado pelo WFMS em um evento especificado que ocorre durante uma execução de processo. Com lógica de programação nós queremos dizer um pedaço de software em formato binário ou source, em qualquer linguagem de programação ou de script. A camada de lógica de programação combina todos esses trechos de software com a informação que especifica em quais eventos esses trechos devem ser executados. Alguns exemplos de lógica de programação são enviar um e-mail, atualizar o banco de dados, obter dados de algum sistema.

A camada da interface de usuário

Quando um ator sinaliza o fim de um estado, geralmente este é o evento onde os dados são colocados nas variáveis de processo. Voltando ao exemplo do pedido de férias (pág. 4), quando o chefe sinaliza ao WFMS que o estado ‘avaliação’ está feito, o chefe fornece o valor ‘aprovado’ ou ‘reprovado’ ao processo. Alguns WFMS’s permitem especificar para cada estado quais dados podem ser inseridos no processo e como eles devem ser armazenados nas variáveis de processo. A partir dessa informação, um formulário da interface de usuário pode ser gerado para sinalizar que requer informação do usuário.

As implementações de workflow

Projetos open source

Até aqui foi dada uma idéia geral do que é e como funciona um WFMS. A seguir vem uma lista de sistemas reais que gerenciam workflows. A difícil tarefa aqui é decidir qual se adapta melhor às suas necessidades.

- jBpm
- OpenEbXML
- Werkflow
- OSWorkflow
- wfmOpen
- OFBiz
- ObjectWeb Bonita
- Bigbross Bossa
- XFlow
- Taverna
- Enhydra Shark
- PowerFolder
- Breeze
- Open Business Engine
- OpenWFE
- Freefluo
- ZBuilder
- Twister
- Con:cern

Vendedores comerciais

- Bea's WLI
- Carnot
- Dralasoft
- Filenet
- Fujitsu's i-Flow
- IBM's holosofx tool
- Intalio
- Lombardi
- Oakgrove's reactor
- Oracle's integration platform
- Q-Link
- SAP's NetWeaver
- Savvion
- Seebeyond
- Sonic's orchestration server
- Staffware
- Ultimus

- Versata
- WebMethod's process modeling

Especificações

O site a seguir é uma apresentação feita por Michael zur Muehlen sobre as especificações de workflow - http://www.wfmc.org/standards/docs/Process_Management_Standards_files/frame.htm.

Muitos autores acreditam que os padrões definidos para workflow ainda estão se desenvolvendo. Um argumento para isso é a grande quantidade de especificações que se sobrepõem hoje em dia.

De acordo com Tom Baeyens, a enorme quantidade de especificações e limitada adoção de cada uma delas são devido ao limitado conhecimento sobre as primitivas. Workflow é um assunto que você se atrapalha facilmente. A distração ocorre quando os conceitos de workflow se misturam com outras tecnologias ou conceitos. Um exemplo bastante concreto: workflow é completamente complementar a web-services. Você poderia fornecer uma interface para acessar um WFMS como um web-service, mas definitivamente é uma má idéia assumir que um WFMS sempre tem que ser acessado através de uma interface web-service. Algumas especificações assumem esse fato. Além de web-services, outras distrações freqüentes ocorrem com e-mail, aplicações web e estrutura organizacional,...

A primeira tentativa de padronização em workflow foi a Workflow Management Coalition (WfMC). A WfMC começou em 1993. Uma publicação muito interessante dessa Coalizão é o modelo de referência. Ele define as interfaces entre um sistema gerenciador de workflow e outros atores. Outro trabalho é a especificação XPDL. XPDL define um schema XML para especificar a parte declarativa do workflow.

- JSR 207: Process Definition for Java
- WfMC's XPDL
- ebXML's
- BPMI's BPML & WSCI
- BPEL – Uma proposta dessa especificação para Java está sendo feita: BPELJ
- OMG's Workflow management facility
- UML
- RosettaNet
- UBL

A utilização do jBPM

Como foi mostrado, existem vários sistemas capazes de gerenciar workflows. A escolha do jBPM foi devido a algumas de suas características:

- **Open source** – por ser código aberto e com licença LGPL permite que o usemos em nossa aplicação sem ter que gastar qualquer quantia com compra ou licenciamento de software.

- **Linguagem Java** – ser implementado em Java também foi levado em consideração no momento da escolha, já que todo o sistema onde o jBPM é utilizado também é desenvolvido nessa linguagem, facilitando assim sua integração.

- **Editor gráfico** – embora não tenhamos utilizado o seu editor de processos de workflow, que é fornecido como plug-in para o Eclipse, foi a partir dele que construímos o nosso próprio editor para que funcionasse na interface web.

- **Estabilidade/Suporte** – mesmo estando em constante aprimoramento, já está numa versão estável, de forma que, se ocorrer algum erro, é possível encontrar alguma ajuda nos fóruns.

- **MySQL** – o sistema ConteXpress foi desenvolvido utilizando a base de dados MySQL, então o fato do jBPM suportá-lo para armazenar seus dados foi mais um ponto positivo na hora da escolha do WFMS.

Com essas características o jBPM satisfez nossas necessidades já que era uma ferramenta capaz de rodar em uma aplicação web, controlar o fluxo dos documentos, atribuir tarefas a usuários, permitir a criação da definição de processo através de um editor gráfico em uma interface web, além de ser uma ferramenta que é desenvolvida em Java e sua relação custo/benefício valer a pena.

Tom Baeyens acredita que um workflow deve suportar uma combinação de 1 - especificação declarativa do estado de um workflow e 2 - lógica de programação. O jBPM é desenvolvido a partir

de dois princípios. Primeiro, ele fornece um mecanismo muito simples para se implantar uma máquina de estados, fazendo com que fique fácil para os desenvolvedores Java incorporarem jBPM em seus projetos. Segundo, é desenvolvido para permitir flexibilidade aos padrões mais complexos de workflow.

Outras ferramentas utilizadas

Struts/JSP

Como o sistema ConteXpress já era baseando na arquitetura MVC (Model-View-Controller) onde, de um modo geral, o framework Struts representa o Controller e as JSP's (JavaServer Pages) representam o View. A parte de workflow também foi adaptada para interagir com o sistema dessa forma.

O Struts facilita a interação usuário/sistema através de seus Action's e Form's permitindo ao usuário realizar suas tarefas na interface web (JSP's) e, através das funcionalidades do framework, elas serem executadas no sistema.

Eclipse IDE

Por já ser utilizado e conhecido, esse foi o IDE escolhido para o desenvolvimento da solução. Além disso, o Eclipse possui várias funcionalidades que facilitam a programação tornando-a mais dinâmica.

Outro fator que nos levou a continuar usando-o, foi o fato de que o jBPM possui um plug-in para ele que permite a criação de processos através de um editor gráfico. Esse editor foi usado como modelo para que pudéssemos construir nosso próprio editor que funcionasse em ambiente web.

MySQL/Hibernate

Base de dados já utilizada no ConteXpress e suportada pelo jBPM, o MySQL foi utilizado para armazenar os dados dos processos como definições de processos, instâncias de processos, suas

variáveis, estados, tarefas e qualquer tipo de informação que precisava ser persistida na base para eventual utilização futura.

Para fazer a persistência, o jBPM utiliza internamente o **Hibernate**, por isso, também foi necessário um conhecimento mínimo sobre o mesmo para saber como funciona a busca dos dados e suas inicializações (lazy initialization).

Capítulo 3 – Atividades realizadas

Neste capítulo descreveremos todo o estudo e trabalho realizados no estágio na Murah Technologies durante o período em que estive dedicado ao desenvolvimento da atualização e implantação do sistema que gerencia um fluxo de trabalho.

Estudo da versão antiga

Após ler documentos impressos e artigos na web (ver bibliografia) para me familiarizar com o contexto de WFMS's, passei para a parte mais prática onde eu teria que atualizar o sistema de workflow que já estava integrado com o ConteXpress. Para saber por onde começar a atualização, tive que estudar a parte de workflow do sistema, que é um praticamente um projeto a parte.

Mas antes disso, foi necessário compreender o funcionamento do framework Struts e das JSP's para que eu pudesse acompanhar o fluxo como um todo, desde a ação escolhida pelo usuário na interface web até a sua execução no workflow, passando pelo controle do Struts.

Com o estudo desse framework compreendi que é possível manter um controle sobre as ações de uma JSP através de Action's e Form's, de forma que uma classe em Java que represente uma Action seja responsável pela execução de alguma ação, no caso chamaria os métodos necessários para o funcionamento do workflow, e aquela que representa um Form seja responsável pelos atributos utilizados na JSP e suas validações. Além disso, existem arquivos de configuração, onde é necessário fazer a associação entre a JSP, o Action e o Form.

Feito esse estudo, pude compreender melhor o funcionamento da parte de workflow do sistema. Para realizar esse segundo estudo utilizei o código implementado da versão antiga (jBPM 2.0), fóruns e User Guide disponíveis no site do jBPM e algumas pesquisas no Google também foram necessárias para buscar informações e até mesmo o código fonte da versão 2.0. Com esse material foi possível começar a entender como estava implementado o workflow no sistema, mas só foi possível o real entendimento quando fui fazer a substituição das versões do jBPM, pois nesse momento não foi uma simples substituição de bibliotecas (arquivos .jar), pois haviam métodos, e até mesmo classes, que não existiam mais, ou que passaram a existir, na versão recente (jBPM 3.1).

Atualização jBPM 2.0 para 3.1

Após baixar a versão 3.1 do JBoss jBPM Starters Kit, dei início à substituição das bibliotecas. Substituindo o arquivo núcleo do jBPM (jbpm-3.1.1.jar) e as bibliotecas externas necessárias (hibernate3.jar, antlr-2.7.5H3.jar,...) já apareceram alguns erros de compilação identificados pelo IDE Eclipse. Foi nesse ponto que o entendimento sobre o funcionamento do workflow se aprofundou, pois para reparar esses erros de compilação tive que entender qual o contexto no qual estava inserida aquela parte do código na versão 2.0 e saber como desempenhar aquela mesma função na versão recente.

Além disso, a versão 3.1 do jBPM possui tabelas diferentes da versão antiga para armazenar os dados. Por isso, comecei a trabalhar também com o MySQL, base de dados já utilizada no sistema e suportada pelo jBPM. Além das bibliotecas, o Starters Kit possui também um pacote para base de dados compatíveis que contém toda informação, drivers e scripts necessários para fazer o jBPM rodar em uma determinada base de dados. Utilizando esse pacote, foi possível gerar scripts que criam e apagam as tabelas a partir de configurações determinadas em um arquivo.

Com os erros de compilação corrigidos e as tabelas do banco de dados atualizadas pude passar para a parte de testes. Durante a execução desses, mais alguns erros ocorreram devido à incompleta compreensão dos conceitos envolvidos. Com mais um pouco de aprendizado e correções feitas, pude finalmente empacotar (gerar um arquivo .zip) o código gerado com a versão 3.1, já que este ainda não seria implantado.

A utilização da versão 3.1

Inicialmente o objetivo do trabalho era verificar a viabilidade da atualização da versão 2.0 do jBPM para a versão 3.1 gerando um pacote com o código atualizado para futura utilização, mas acabamos tendo que utilizar a nova versão.

Com o código implantado, tivemos que corrigir alguns erros de execução para que funcionasse como na versão antiga. Essas correções às vezes eram trabalhosas devido ao projeto utilizado na versão 2.0. Assim, foi decido refazer a parte de workflow do sistema utilizando a versão 3.1.

Refizemos essa parte tendo em mente as funcionalidades que o workflow deveria possuir, procurando ao máximo não utilizar o que já estava feito. Apenas algumas idéias e códigos foram reutilizados. Dessa maneira, apesar de perdermos algum tempo reconstruindo o workflow, ficou mais fácil de trabalhar com ele. Por exemplo, a parte de criação de uma definição de processo era feita em uma tela com tabelas e boxes para representar os estados e atores, agora é feita em um editor gráfico; outro exemplo é que a o ConteXpress pode se comunicar com o workflow através de uma classe que serve de fachada para os métodos que executam os métodos do jBPM, ficando transparente para outros usuários (programadores).

Essa parte do trabalho foi realizada numa espécie de programação pareada com o Nelson Takashi Omori, fato que proporcionou um desenvolvimento mais rápido. Primeiramente, nós implantamos a nova versão e depois, refizemos o código. Com o núcleo do código refeito o Nelson dedicou-se ao editor gráfico para definições de processos e eu continuei aperfeiçoando o código do workflow propriamente dito.

Atualmente a versão 3.1 já está implantada, mas está em constante aprimoramento.

Capítulo 4 – Resultados obtidos

Os resultados do trabalho podem ser divididos em duas fases: a primeira, onde tive que gerar um pacote com o código do sistema ConteXpress com a versão 3.1 do jBPM, mostrando a viabilidade de sua atualização; e a segunda, onde implantamos o código realmente, criando um novo WFMS.

Na primeira fase, eu já havia estudado sobre workflow e seus gerenciadores, focando o jBPM, mas foi durante a segunda fase que os conceitos se fixaram, pois durante a implantação foi necessário compreender melhor como funcionava a execução do processo, desde a criação da definição até a realização da última tarefa, passando por ações que devem ser executadas. Além disso, o resultado obtido após a implantação da nova versão do jBPM não foi muito baseado no pacote gerado, já que nós preferimos refazer a parte de workflow, mas o resultado da primeira fase serviu como base para entendermos como deveria funcionar o novo WFMS.

Portanto, pode-se dizer que o resultado da primeira fase foi útil tanto para compreendermos os conceitos envolvidos, como para servir de ligação entre as versões 2.0 e 3.1 do jBPM, para que assim, pudéssemos criar o produto final que foi o novo WFMS, que possui algumas melhorias claras:

- A criação da definição de processo não é mais feita através de página com tabelas, aonde o usuário escolhe os estados e atores responsáveis, e sim, através de um editor gráfico.

- A parte do workflow no ConteXpress ficou mais independente, como um projeto a parte, já que possui uma espécie de fachada para chamar os métodos do jBPM e os métodos utilizados para acessar as funcionalidades do jBPM estão menos dependentes do resto do sistema.

- Novas funcionalidades como: desabilitar uma determinada definição de processo, escolher qual definição deseja utilizar, adicionar um timer no estado para que quando terminar o tempo desse timer o processo continue, por exemplo.

Capítulo 5 – Conclusão

A respeito de workflow e seus gerenciadores, a conclusão que se pode tirar é que:

1. Workflow é como uma máquina de estados onde se tem um estado inicial para começar o processo, os estados intermediários, que são as tarefas, os estados que se bifurcam (fork's) para execuções concorrentes e outros tipos de estados; e o estado final, onde termina o processo.

2. Para iniciar o processo é necessária uma definição de processo que define os estados, atores, ações,... Essas definições têm que seguir especificações.

3. As especificações ainda estão muito fracas, nenhum padrão foi amplamente definido nessa área.

4. Por isso, existem muitas ferramentas de workflow e decidir qual delas é a melhor para se utilizar, talvez seja a tarefa mais difícil para as empresas que querem adotar as idéias de BPM.

Apesar de ainda faltar uma melhor padronização, já existem bons sistemas gerenciadores de workflow. E isso já é um grande incentivo para que as empresas comecem a utilizar essa ferramenta.

Falando do jBPM mais especificamente, ele uma dessas ferramentas de WFMS que existem no mercado hoje em dia. Através dele temos uma interface de alto nível para realizar as funções de um workflow. Algumas de suas características são:

- Apesar de não possuir uma documentação completa, sua utilização para implementar os processos pedidos pelos usuários é simples, tendo-se os conceitos claros.

- Ele permite que os usuários e os desenvolvedores se entendam devido ao modo como define os elementos.

- É open source!

Finalmente, sobre a parte prática, além de aprender novos conceitos e novas ferramentas (workflow, MySQL, STRUTS,...), alcançamos nossos objetivos: atualizar o código existente para a versão 3.1 e construir um WFMS com mais recursos para o usuário e melhor estruturado, facilitando o entendimento para o desenvolvedor.

Capítulo 6 – Bibliografia

---

[1] JBoss jBPM – White Paper by John Koenig
http://jboss.com/pdf/jbpm_whitepaper.pdf

[2] The State of Workflow by Tom Baeyens
http://jboss.com/products/jbpm/stateofworkflow

[3] JBoss jBPM User Guide
http://docs.jboss.com/jbpm/v3/userguide/

[4] Struts
http://struts.apache.org/

[5] JSP
http://java.sun.com/products/jsp/

[6] Eclipse
http://www.eclipse.org/

[7] MySQL
http://www.mysql.com/

[8] Hibernate
http://www.hibernate.org/

Parte II – Parte Subjetiva

Capítulo 1 – Desafios e frustrações

Dentre as atividades realizadas no estágio na empresa Murah Technologies, é difícil identificar uma em especial que tenha sido mais desafiadora. Isso porque todas elas exigiram bastante dedicação, desde o estudo sobre o assunto até o desenvolvimento da nova ferramenta de workflow, já que se tratava de um assunto que eu nunca tinha visto antes, mesmo parecendo um tanto quanto intuitivo.

A parte dos estudos pode ser considerada desafiadora pois envolveu muitos conceitos novos, que foram além de workflow e BPM. Tive que compreender o funcionamento do framework Struts, ter noções do Hibernate e saber utilizar as JSP's, ou seja, foi uma fase do trabalho na qual houve uma grande quantidade de conhecimento que só pode ser absorvida quando fui para a parte prática.

Quanto à implementação, o desafio foi maior. Afinal, foi necessário colocar todo aquele aprendizado em prática, e, muitas vezes, tive que voltar à bibliografia para esclarecer alguns conceitos. Enquanto implantava a nova versão, pude assimilar melhor o funcionamento do workflow no sistema ConteXpress. O desafio nessa parte foi trabalhar com projetos de complexidade muito maior que os vistos na faculdade.

Finalmente, o último grande desafio foi reescrever o código do workflow, desenvolvendo um novo WFMS. Nós já havíamos atualizado uma boa parte do código, mas o código não estava fácil de ser alterado. Então, tomar a decisão de recomeçar não foi tão simples, já que exigiria um tempo para chegarmos ao mesmo ponto em que

estávamos. Mesmo assim achamos melhor recomeçar e realmente ficou melhor.

As frustrações não foram muitas, mas entre elas está principalmente o fato de a documentação (Javadoc) não estar completa e o User Guide nem sempre nos ajudar. Às vezes, os fóruns resolviam nossos problemas, mas algumas vezes olhamos no código fonte do jBPM. Outro fato é que inicialmente o objetivo era simplesmente gerar uma versão atualizada do sistema, mas sem implantá-la efetivamente, mas acabamos tendo que utilizar a nova versão e é gratificante ver algo funcionando.

Capítulo 2 – Matérias mais importantes do BCC no trabalho

Primeiramente vale ressaltar a importância do curso como um todo, pois quando entrei na faculdade eu não sabia nada de programação e após 3 anos de curso consegui ter uma boa base para entrar no mercado de trabalho. Então, acredito que todas as matérias tiveram sua importância, seja direta ou indiretamente, ajudando na implementação ou no entendimento dos conceitos.

Mesmo assim, algumas matérias se fizeram mais presentes durante o desenvolvimento do trabalho:

- **Introdução à Computação e Princípios de Desenvolvimento de Algoritmos**: por serem matérias básicas, acredito que foram importantes não só para o estágio, mas também para a minha formação como um todo.

- **Estruturas de Dados**: É fundamental para qualquer projeto saber como armazenar os dados que se possui de forma que fique fácil a sua manipulação.

- **Conceitos Fundamentais de Linguagens de Programação**: sua importância no trabalho foi principalmente por introduzir o conceito de avaliação “adiada”, onde os dados são buscados somente quando requeridos. Muito utilizado na compreensão das consultas do jBPM.

- **Algoritmos em Grafos e Linguagens Formais e Autômatos**: duas matérias que acabaram me auxiliando no entendimento dos conceitos de workflow, já que esse é como uma máquina de estados. Então os conceitos vistos nas matérias frequentemente puderam ser aplicados no projeto.

- **Sistemas de Bancos de Dados**: importante disciplina por apresentar os fundamentos das consultas SQL. Assim, quando foi necessário criar tabelas e realizar consultas utilizando o MySQL, a tarefa ficou mais fácil.

- **Engenharia de Software**: matéria na qual aprendi conceitos mais voltados para o desenvolvimento prático, técnicas de boa programação, ou seja, conceitos que podem ser aplicados em qualquer projeto, bastando adaptar às condições em que se encontra.

- **Programação Orientada a Objetos**: apesar de não ter terminado de cursar essa matéria esse ano, acredito que ela seja importante, pois durante o estágio trabalhei com alguns padrões e Java. Pretendo cursá-la ano que vem.

Capítulo 3 – Interação com o responsável e com a empresa

Desde que cheguei à Murah Technologies, o ambiente de trabalho tem sido muito bom. Apesar de haver uma distinção hierárquica para efeito de estrutura empresarial, na prática é possível se relacionar com todos tranquilamente e de maneira descontraída. Outro fator que favoreceu o bom clima na empresa, foi o fato de que, mesmo sendo estagiário, as idéias e sugestões serem sempre bem vindas nas reuniões e no desenvolvimento do sistema.

O relacionamento com o Fabrício Garcia Imbrizi, responsável pelo ConteXpress, não é diferente, sempre disposto a ouvir e dar sugestões, tem sido muito importante para essa minha experiência profissional. Além dele, o Nelson também faz parte da equipe, com ele a comunicação é mais freqüente, pois estamos sempre desenvolvendo e precisamos saber como está o andamento do projeto ou tirar dúvidas. Assim, o relacionamento ficou parecido com aquele nos trabalhos da faculdade, onde é possível se relacionar com todos de uma maneira descontraída, ainda mais que o Nelson também é do IME-USP, talvez a maior diferença seja que a responsabilidade é maior.

Apesar de toda essa flexibilidade, até mesmo nos horários, a responsabilidade, o cumprimento dos prazos e a produtividade são uns dos primeiros itens na lista de prioridades da Murah. Isso tudo permitiu um que o ambiente não se tornasse cansativo.

Capítulo 4 – Aprimoração

Mesmo já tendo melhorado consideravelmente, ainda é possível aperfeiçoar a parte de workflow do ConteXpress. Para isso, além de completar a documentação, seria interessante continuar refatorando o código e estudando as funcionalidades oferecidas pelo jBPM, pois algumas vezes já escrevemos códigos mais complicados do que o necessário por não compreender corretamente, ou completamente, o funcionamento dessa ferramenta. Observar as novas versões do jBPM também é recomendável, para ver se uma nova atualização vale a pena. Podemos ainda oferecer mais funcionalidades. Ou seja, é um trabalho em constante desenvolvimento.