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, fevereiro de 2007

Sumário

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. Devido à necessidade o trabalho se estendeu e chegamos a implantar a nova versão.

Antes de começar essa pequena abordagem sobre workflow e seus gerenciadores, irei comentar um pouco sobre dois termos freqüentemente confundidos nessa área, mas que são conceitualmente diferentes: workflow e Business Process Management (BPM). O primeiro, bem mais utilizado, tanto entre programadores quanto entre usuários, refere-se àquilo que o usuário vê, às interfaces do sistema, ou seja, serve para denotar a parte prática de todo o conjunto de operações envolvidas nessa interação entre o usuário e o programa. Já o BPM vai além das interfaces, ele engloba toda a estrutura necessária para que o workflow possa funcionar. Assim, ele abrange desde assuntos empresariais até os

assuntos relativos ao workflow propriamente dito. Nesse trabalho discutirei mais sobre workflow e os sistemas que os gerenciam.

Problemas relacionados ao 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.

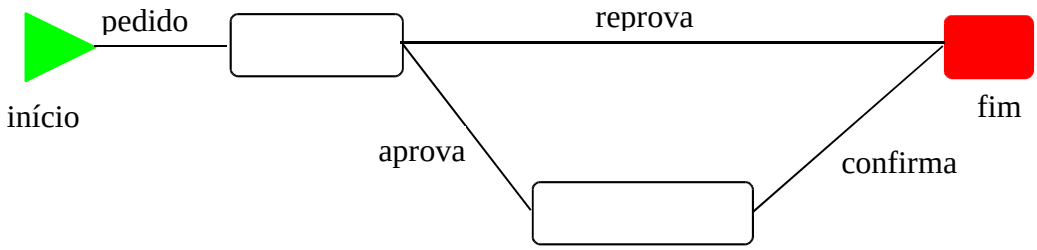

Figura 1: Exemplo de fluxo de trabalho: pedido de férias.

O que é um Workflow Management System (WFMS)

Definições

Um workflow management system (WFMS), como o próprio nome diz, serve para gerenciar processos de workflow, mas para conseguir fazer isso ele precisa ter conhecimento de como é o fluxo do processo, qual é o sentido que as ações podem tomar. Além disso, ele precisa manter as informações de cada execução de um determinado processo, para que assim os dados de cada estado possam ser utilizados em passos futuros e, conseqüentemente, pessoas e aplicações possam realizar suas tarefas.

Temos, portanto, três componentes que são fundamentais em um WFMS. O primeiro é a definição de processo, responsável pela

descrição formal do fluxo do processo dita anteriormente, permitindo que o gerenciador saiba quais estados estão participando do processo e como eles estão relacionados, essa descrição será vista mais a fundo adiante. O próximo componente é a instância de processo que representa uma execução dessa definição formal, é onde os usuários e sistemas podem agir. O terceiro componente é a variável de contexto que armazena os dados em uma instância de processo para que os usuários e sistemas possam reaproveitá-los quando necessário em situações futuras na mesma instância de processo.

Apesar da necessidade de haver uma descrição formal, o modo como ela deve ser feita ainda não está muito bem definido. Como veremos mais no Capítulo 2, ainda há muita divergência nessa área.

Casos de uso

Considerando os exemplos estudados sobre workflow, acredito que eles possam ser divididos em três grandes grupos de casos de uso simplesmente descritos aqui, sem qualquer tipo de ordenação.

Uma primeira opção seria utilizar um WFMS para fazer uma integração entre aplicações, criando uma enterprise application integration (EAI). Assim, cada aplicação continuaria desempenhando seu papel normalmente, mas com o gerenciador de workflow, ela seria chamada assim que necessário e em seguida o fluxo poderia continuar. Por exemplo, atualmente as empresas possuem aplicações para gerenciar clientes, entrada de pedidos, emissão de contas, e outras tarefas. Com o WFMS, a aplicação de entrada de pedidos poderia ser executada, guardando eventuais dados para serem utilizados na aplicação de gerenciamento de clientes, para que finalmente a aplicação de emissão de contas pudesse ser executada.

O segundo grupo que se destaca são os exemplos que utilizam WFMS para desenvolver software de workflow com tarefas relacionadas às pessoas. São bastante utilizados por empresas que desejam certificados, já que permitem a automação, através das definições de processo, dos procedimentos que antes eram escritos.

A última opção é criar uma aplicação onde um dos seus componentes seja o workflow. Ou seja, é como se uma daquelas aplicações mencionadas no primeiro exemplo (emissão de contas, entrada de pedidos, gerenciamento de clientes) possuísse uma ferramenta de workflow. A vantagem de se fazer isso é que facilita a

manutenção do programa. Foi principalmente nesse grupo que desenvolvemos o nosso trabalho, já que adicionamos um componente de workflow no sistema que já existia.

Argumentos para o workflow

Apesar das empresas de um modo geral ainda terem receio de introduzir workflow, a sua implantação traz muitos benefícios em vários aspectos.

- **Desenvolvimento fica facilitado** – 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.

- **Aumenta o controle do processo** - diminui as chances de atrasar as tarefas e perder documentos, já que será tudo controlado pelo gerenciador.

- **Diminui o custo de gerenciamento** – uma vez que os administradores terão mais controle sobre suas tarefas, poderão efetuá-las mais rapidamente, podendo aproveitar o tempo que era gasto com esse gerenciamento para aperfeiçoar o trabalho dos funcionários ou outras coisas.

- **Aumento da produtividade** – realizando as tarefas mais rapidamente, designadas para as pessoas melhor qualificadas, é possível aumentar a produtividade, podendo equilibrar os gastos.

- **Segurança e Privacidade** – Somente as pessoas autorizadas terão acesso aos dados e como o gerenciador mantém o controle sobre qual o estado atual do processo e quais as ações tomadas pelos atores, torna-se bem mais difícil os dados se perderem.

Business process management (BPM)

Como foi visto anteriormente, um dos componentes principais de um WFMS é a definição do processo. Para que você possa criá-la, é

necessário que se tenha bem definido como funciona o processo que se deseja automatizar. Algumas ferramentas fornecem interfaces gráficas para a criação das definições, facilitando a tradução do processo do mundo real para o workflow, mas mesmo assim é necessário analisar o processo antes de criá-lo. O site http://e-workflow.org fez uma descrição das vantagens ganhadas com essa análise e aqui vão alguns exemplos:

- **Maior eficiência** – etapas que não são mais necessárias serão eliminadas conforme os processos são analisados e automatizados.

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

- **Melhor serviço ao cliente** – com o processo reestruturado e melhor controlado a resposta ao cliente será mais rápida trazendo maior satisfação.

Capítulo 2 – Conceitos e tecnologias estudadas

Um olhar mais aprofundado

Interfaces WFMS

Já vimos que um WFMS necessita de uma definição de processo para que possa iniciar uma nova execução e assim os usuários e sistemas possam interagir com o processo. A partir do momento que

é iniciada a execução, o seu contexto é gerenciado para que seja possível utilizá-lo em etapas seguintes e nas trilhas de auditoria (logs), que podem ser acessadas pelo responsável do processo. Através desse funcionamento básico, foram definidas algumas interfaces que servem de modelo para qualquer sistema gerenciador de workflow:

- **Interface 1:** A interface 1 é responsável pela disponibilização das definições de processo em um WFMS. Ela permite que o módulo de criação das definições seja independente do workflow, fazendo com que uma ferramenta de criação possa ser usada em várias engines.

- **Interface 2:** Com o intuito de facilitar a utilização de várias aplicações com o gerenciador de workflow, a interface 2 faz a comunicação entre essas aplicações do cliente e o workflow, indicando a realização de uma tarefa. Ou seja, permite aos usuários e sistemas agirem nas instâncias de processos, execuções de definições de processo.

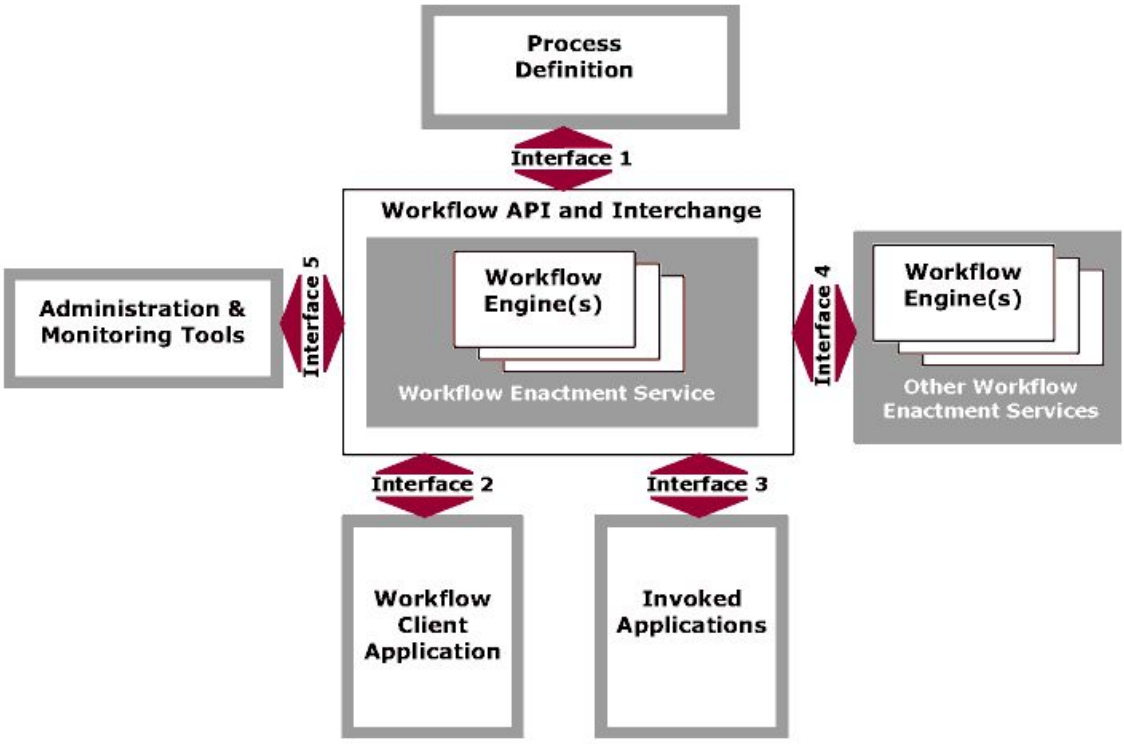

Figura 2: Interfaces de um WFMS.

- **Interface 3:** Ao contrário da interface 2, onde as aplicações chamam o workflow, essa interface representa a comunicação entre o workflow e as aplicações que são chamadas por ele. Por

exemplo, em um determinado passo de uma execução de processo, pode estar definido um evento e esse evento pode ter como ação um código que mande executar algum programa para realizar alguma atualização. Na prática, quando um vendedor confirma a venda poderia mandar imprimir a nota automaticamente.

- **Interface 4:** A interface 4 é destinada à comunicação com outros sistemas gerenciadores de workflow, para que assim também seja possível ter acesso ao núcleo do sistema, podendo ter controle do andamento do processo mesmo que ele tenha sido iniciado em outra engine de outra empresa.

- **Interface 5:** A última interface serve para gerar as trilhas de auditoria (logs) que são usadas pelos administradores para colher estatísticas dos processos. Aqui são gerados históricos tanto das instâncias de processos quanto dos processos de um modo geral. Para as instâncias, pode-se guardar quem realizou a tarefa, quando foi realizada, etc. Para os processos, a opção é tirar uma média desses dados para ter uma idéia como um todo.

As 4 camadas de uma definição de processo

Agora que a estrutura geral de um WFMS já foi descrita, vamos nos aprofundar um pouco na estrutura de um de seus principais componentes: a definição de processo. As definições não possuem um formato padrão, como veremos, existem várias especificações para elas, mas a idéia da Interface 1 (ver Figura 2) é justamente essa, permitir que várias definições possam ser integradas em um gerenciador. O conteúdo de uma definição de processo pode ser dividido em 4 camadas: estado, contexto, lógica de programação e interface de usuário.

A camada de estado

A camada de estado serve simplesmente para especificar os estados que participam do processo e o fluxo que pode ser tomado a partir do momento que se inicia uma execução.

Um estado, geralmente conhecido como wait-state ou node, especifica a dependência em relação a um ator externo. De um modo

geral, o sentido de um estado em um processo seria “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.” Além disso, os estados podem possuir várias especializações cada uma representando um tipo específico de atividade a ser realizada. Alguns exemplos comuns de estados são: node, fork, join, state. Um exemplo típico de estado é um passo de aprovação.

Uma característica importante dos WFMS’s é a possibilidade de definir os responsáveis pelas tarefas em determinados estados. Essa atribuição de tarefas permite a criação das swimlanes, que nada mais é do que a lista dos atores que podem executar as tarefas daquele estado. Com isso, os gerenciadores conseguem determinar as listas de tarefas a serem executadas e como os atores também podem ser pessoas, além de programas, é necessário um cálculo em tempo de execução para verificar se uma pessoa tem permissão para realizar tal tarefa.

O fluxo de uma definição de processo determina o relacionamento entre os estados especificados. Para determinar esse fluxo são utilizadas as transições e os tipos de estado que, como foi dito acima, podem possuir várias especializações, e de acordo com a sua semântica podem ser usados para um ou para outro fluxo. Por exemplo, caminhos concorrentes são modelados com forks e joins, enquanto que caminhos exclusivos são modelados com decisions, states e nodes.

Quando se inicia uma nova execução da definição de processo, o WFMS cria um token para que ele possa identificar em qual estado do processo a execução está. Dessa maneira, é possível contabilizar o tempo gasto tanto em um determinado passo quanto na execução inteira, auxiliando na geração das estatísticas. Além disso, sabendo-se qual o estado atual, o gerenciador consegue saber qual tarefa está sendo executada. É também nesse token que, após consultar os dados necessários, é definido o ator que irá executar essa tarefa, podendo assim obter a lista de tarefas.

A camada de contexto

É nessa camada que são declaradas as variáveis de contexto. Elas servem para armazenar dados de uma determinada instância de processo. Assim, apesar de ser declarada na definição de processo, cada execução do processo possuirá seus próprios valores. Esses

valores podem ser pré-definidos ou podem ser definidos pelo programador, depende do WFMS.

A utilização dessas variáveis é diversa, elas podem ser usadas para guardar alguma informação do próprio processo que seja necessária em um estado futuro, mas pode também guardar informações externas ao processo, como um índice no banco de dados, por exemplo, basta tomar os devidos cuidados para mantê-las atualizadas.

A camada de lógica de programação

Os gerenciadores permitem que eventos ocorram em determinadas situações (entrada e saída dos estados, nas transições de um estado para o outro) e esses eventos podem ter ações, que são trechos de código em alguma linguagem de programação ou de script. Essa camada é responsável por determinar onde e quando esses trechos de código serão executados. Alguns exemplos práticos são o envio de e-mail para o(s) responsável(is) assim que o processo acaba de entrar em um estado, a atualização do banco de dados quando sai de um estado.

```
<process-definition>
  <start-state>
    <transition to='s' />
  </start-state>
  <state name='s'>
    <transition to='end'>
      <action class='org.jbpm.tutorial.action.MyActionHandler' />
    </transition>
  </state>
  <end-state name='end' />
</process-definition>
```

Figura 3: Exemplo de definição de processo seguindo a especificação do jBPM (JPDL – jBPM Process Definition Language).

A camada de interface de usuário

Essa é a camada onde o usuário, ator da tarefa, pode inserir dados no processo. Como as variáveis de contexto costumam ser gravadas na saída dos estados, os gerenciadores de workflow podem fornecer interfaces com formulários para que o usuário selecione quais dados ele gostaria de armazenar nas variáveis, ou ainda, quais dados ele gostaria de adicionar. A liberdade com que se pode

adicionar ou selecionar os dados que serão colocados nas variáveis varia de gerenciador para gerenciador.

Algumas implementações de workflow

Agora que já foi dada uma idéia geral de como funciona um WFMS e do que ele é constituído, vou descrever alguns exemplos de implementações vistas durante o tempo em que estivemos pesquisando. A maioria delas, se não todas, é desenvolvida em Java, porém algumas possuem versões em outras linguagens. Para os projetos open source uma boa referência é [10]. Os sites foram visitados em 08/02/2007.

Projetos open source

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

Vendedores comerciais

- Bea's WLI
- Carnot
- Dralasoft
- Filenet
- Fujitsu's i-Flow
- IBM's holosofx tool
- Aspose.Workflow
- Sonic's orchestration server
- Ultimus

Especificações

Como vimos, um WFMS deve ter uma descrição formal do processo para conseguir manter o controle sobre cada instância do processo. Essas descrições têm uma estrutura geral, mas, para que o gerenciador possa compreendê-las, elas devem seguir especificações.

Uma das grandes diferenças entre os sistemas de workflow está justamente em suas especificações, pois, atualmente, cada WFMS

acaba criando a sua própria especificação, o que gera a enorme quantidade de ferramentas disponíveis e uma certa dificuldade de integração entre elas.

Alguns autores acreditam que a baixa adoção das especificações, e conseqüentemente das ferramentas, pode ser devido à falta de conhecimento dos fundamentos do assunto e, assim, ao invés de usarem alguma ferramenta já existente, acabam criando mais uma, que talvez também não tenha a semântica correta.

Pensando nisso, em 1993, criou-se um grupo para tentar definir alguns padrões para a área de workflow: a Workflow Management Coalition (WfMC). A Coalizão foi dividida em grupos de trabalho, sendo que cada grupo ficou responsável por desenvolver uma parte dessa padronização. Por exemplo, existem grupos para cada uma das cinco interfaces de um WFMS descritas anteriormente. Essa Coalizão já fez publicações como o modelo de referência e a especificação XPDL que são bastante utilizadas como base de estudo para desenvolvedores de workflow.

A seguir estão algumas das especificações que podem ser encontradas hoje em dia:

- JSR 207: Process Definition for Java
- WfMC's XPDL
- ebXML's
- 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.

Através do jBPM foi relativamente simples de se implantar um workflow no nosso sistema, tanto pela sua API, apesar de não ser tão bem documentada, quanto pela sua flexibilidade, pois já tivemos que utilizá-lo com outra base de dados que não o MySQL.

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, as conclusões 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, possivelmente, 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 e o fluxo entre eles, os atores, as ações,... Elas têm basicamente 4 camadas: de estado, de contexto, de lógica de programação e de interface de usuário. Apesar disso, essas definições têm que seguir especificações.

3. Ainda não há um padrão entre as especificações, então cada um acabou criando a sua.

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 implantar um gerenciador de workflow.

5. Com a evolução da tecnologia a tendência é a padronização, um exemplo disso é a WfMC (1993).

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] Introduction to Workflow - Charles Plesums
http://www.e-workflow.org/downloads/introduction_to_workflow02.pdf

[2] WfMC – Workflow Reference Model
http://www.wfmc.org/standards/referencemodel.htm

[3] WfMC – Workflow Reference Model
http://www.wfmc.org/standards/docs/tc003v11.pdf

[4] The Workflow Reference Model 10 Years On
http://www.wfmc.org/standards/docs/Ref_Model_10_years_on_Hollingsworth.pdf

[5] Transform Magazine - What's the Difference Between Workflow and BPM?
http://www.transformmag.com/showArticle.jhtml?articleID=16400140

[6] Introdução aos Sistemas de Workflow - Jorge Cardoso
http://dme.uma.pt/jcardoso/Teaching/ASI/WorkflowSystems/My%20Lectures/%5B1%5D%20Introducao%20WfMS.pdf

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

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

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

[10] Open Source Workflow Engines in Java
http://java-source.net/open-source/workflow-engines

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

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

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

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

[15] 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 nesse ano.

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.