

Avaliação do uso de ferramentas de workflow em processos típicos de engenharia de software

Walter Itamar Mourão

Arcadian Tecnologia S/A.

Av. do Contorno 3505, sala 802 – Belo Horizonte – MG – CEP: 30110-090

walter@arcadian.com.br

Resumo: Este artigo descreve a seleção de uma ferramenta de workflow e uma experiência de seu uso no processo de desenvolvimento de caso de uso, traçando considerações a respeito das possíveis melhorias no processo que podem ser implementadas com esse tipo de ferramenta.

1 Introdução

Uma vez que a cultura de processo está se disseminando por empresas de todos os portes é natural que aumente o número de ferramentas que apoiem a aplicação das suas melhores práticas. As ferramentas de workflow classificam-se tipicamente nessa categoria.

Uma ferramenta de workflow (WFMS – workflow management system) é um componente de software que recebe como entrada descrições formais de processos de negócio e mantém o estado da execução desses processos, delegando atividades entre as pessoas e aplicações [1].

Apesar dos conceitos funcionais se confundirem constantemente, existem basicamente três tipos dessas ferramentas: frameworks que são embutidos na solução de software específico para a coordenação de processos de software, ferramentas de repasse de mensagens e orquestração de serviços e ferramentas de acompanhamento de processos organizacionais[2]. Estas últimas serão o foco deste trabalho por seu caráter de interface entre o usuário e a definição do processo e sua capacidade documental de servirem como repositórios de processos.

O aspecto de repositório de processos por si só traz grandes benefícios ao dia a dia de qualquer organização pela possibilidade do compartilhamento de conhecimento com ordenação e sem redundâncias e pela capacidade de coleta e reuso de conhecimento[3], no entanto essa característica estática não seria suficiente para justificar a utilização desse tipo de ferramenta, uma vez que esse objetivo pode ser alcançado por ferramentas específicas de gestão de conteúdo ou similares que tem implementação e implantação tipicamente mais simples.

Por outro lado o fato de podermos garantir que o passo a passo de cada processo aconteça da forma prevista na elaboração do processo e que esse acontecimento seja registrado na mesma granularidade em que o processo foi definido deve ser de grande impacto positivo na qualidade do software que está sendo construído pela sua influência em pelo menos três áreas críticas: monitoramento do processo, automatização de integrações e geração de medições.

O monitoramento do processo é um dos requisitos para que se alcance o nível 2 de capacidade do processo, de acordo com a norma ISO/IEC 15504[4]. Uma vez que a ferramenta de workflow apresente claramente a situação atual de cada atividade e qual é o envolvimento atual de cada participante nas atividades atualmente em andamento, têm-se como consequência um quadro atualizado do andamento do processo em questão, facilmente transportável para um gráfico de Gantt, por exemplo.

Entende-se aqui por automatização de integrações tanto a atualização de informações similares entre ferramentas, originalmente não integradas, participantes do processo de desenvolvimento do software quanto o disparo automático de tarefas nessas ferramentas. Como exemplo pode-se citar o momento de entrega de um artefato documental que sinalize simultaneamente o CRM (marcando a tarefa como completa) e o CMS (disponibilizando o artefato para seu público alvo).

O fato das ferramentas de workflow registrarem momentos do processo assim como todas as características interessantes desses momentos abre uma gama de possibilidades de extração de medições uma vez que pode-se usar características nativas de cada atividade como medida básica para uma medição (data e hora do início de uma atividade, por exemplo), como pode-se também acrescentar características que sejam medidas básicas ou derivadas como dados adicionais desses momentos do processo (número de artefatos produzidos, por exemplo). Adicionalmente o fato dessas medições estarem disponíveis imediatamente após a execução da atividade que as originou adiciona a vantagem do uso da medição para avaliação e correção do processo ainda dentro do projeto no qual está ocorrendo sua execução.

Pretende-se aqui apresentar uma avaliação contextual de várias das ferramentas disponíveis atualmente assim como a aplicação simulada de uma delas em um caso prático, com o objetivo de prova dos conceitos apresentados.

1.1 BPM ou workflow

Várias das ferramentas estudadas se intitulam como ferramentas de BPM (Business Process Management) ao invés de usar o termo genérico de workflow. Mesmo que algumas realmente tenham capacidade de gerar simulações ou apoiar a avaliação de riscos dos processos[5] optou-se aqui pela não diferenciação e pela utilização do termo workflow genericamente, uma vez que o objetivo buscado independe da capacidade das ferramentas nesses aspectos.

2 Ferramentas avaliadas

Com o objetivo de selecionar uma ferramenta para o experimento em questão fez-se um levantamento amplo baseado em critérios básicos, seguido da eliminação a partir das dificuldades enfrentadas durante a instalação e execução dos exemplos que as acompanham.

Abaixo são relacionados os critérios básicos para a seleção e sua justificativa:

- Ter sido desenvolvida em Java ou oferecer integração com ferramentas desenvolvidas nessa linguagem.
- As outras ferramentas do ambiente laboratório, tais como Maven[6],

Andromda[7], JxRef[8] e Eclipse[9] foram desenvolvidas em java e uma provável necessidade de integração seria mais facilmente satisfeita.

- Ter o código fonte aberto e não exigir componentes de código fonte fechado.
 - Optou-se por selecionar somente ferramentas cujo código fonte estivesse disponível para consultas ou eventuais customizações.
- Estar disponível em licença de software livre e não exigir componentes de licenças comerciais.
 - Considerou-se que o custo de licença não deveria ser um elemento limitador nesse trabalho, optando-se somente por softwares disponíveis gratuitamente.
- Ser executável nas plataformas Windows e Linux.
 - Considerou-se ser essencial que as ferramentas pudessem ser operadas em ambientes comuns e amplamente difundidos em organizações de todos os portes.
- Ser um ambiente completo com interface com o usuário e persistência implementados de forma nativa.
 - Não se pretendia nesse trabalho desenvolver interfaces gráficas ou outros complementos, salvo interfaces com outras ferramentas já existentes no ambiente.

A partir de pesquisas na internet, em particular nos sites de repositórios de código aberto tais como SourceForge[10] e ObjectWeb[11], e em artigos específicos[1] localizou-se a seguinte lista de ferramentas, com um resumo da descrição dada pelo próprio fornecedor, que atendiam aos critérios básicos:

- Bonita[12]: sistema flexível de workflow compatível com as especificações da Workflow Management Coalition[13] (WfMC), baseado no modelo de workflow proposto pelo ECOO Team[14].
- JBoss jBPM[15]: ferramenta para workflow, BPM e orquestração de serviços que habilita a criação de processos de negócios que coordenam pessoas, aplicações e serviços.
- Nautica[16]: sistema de workflow compatível como modelo proposto pela WfMC e ferramenta de definição no padrão XPD[17].
- OBE – Open Business Engine[18]: mecanismo de workflow flexível, modular e compatível com os padrões WfMC.
- OpenWFE[19]: mecanismo de workflow de código livre.
- OSWorkflow[20]: implementação de workflow de baixo nível.
- PowerStone[21]: sistema de gerenciamento de workflow baseado em Spring[22] e Hibernate[23] composto por um mecanismo, uma console de gerenciamento de fluxo e uma console de gerenciamento de identidade e lista de trabalho.
- Runa WFE[24]: ambiente para workflow/BPM que usa o JBoss jBPM como mecanismo interno.
- Enhydra Shark[25]: servidor de workflow completamente baseado nos padrões da

WfMC e do Object Management Group[26] (OMG) usando XPDL como linguagem de definição nativa.

- Swamp[27]: plataforma de processamento de workflow.
- WfMOpen[28]: implementação de um mecanismo de workflow baseado em Java 2 Enterprise Edition[29] (J2EE) conforme proposto pela WfMC e OMG.
- Yawl[30]: ambiente de workflow criado a partir de rigorosa análise do ambientes de workflow existente.

Dois elementos indicadores da maturidade de um software que estão disponíveis nos primeiros contatos são a sua facilidade de instalação e a qualidade e completude da sua documentação de requisitos e instalação. Sendo assim os primeiros critérios eliminatórios dessas ferramentas foram a facilidade de instalação e qualidade/completude da sua documentação (sendo aceitável que a documentação estivesse em português ou inglês) testados a partir da execução do seu processo de instalação e execução dos exemplos. Adicionalmente foram eliminadas as ferramentas de uso direcionado para situações específicas ou de grande curva de aprendizado. Seguem os produtos eliminados e a justificativa para eliminação:

- Náutica (versão 0.9): documentação em japonês.
- OSWorkflow (versão 2.8.0): várias situações de workflow necessitam de programação manual de scripts. A Interface com o usuário não é configurável e é pouco amigável para o usuário iniciante.
- PowerStone (versão 0.5.1): problemas diversos na conexão com o banco de dados e documentação em chinês.
- Swamp (versão 1.4): problemas de instalação tanto em ambiente Windows quanto em Linux.

Finalmente chegou-se a uma lista de ferramentas que se mostraram a princípio adequadas ao trabalho, contendo ambientes amigáveis de desenvolvimento dos fluxos, de administração e de gerenciamento de atividades. A seguir estão relacionadas estas ferramentas e considerações sobre cada uma, coletadas a partir de sua instalação e execução dos exemplos que as acompanham:

- Bonita (versão 1.7.1): Bastante completa, com um bom painel de gerenciamento das atividades e programa gráfico para criação de fluxos. A comunidade se mostrou bastante ativa na lista de discussões. Além de interface com usuário padrão, conta com interface web baseada no padrão XForms[31] e integração com o portal eXo-platform[32].
- JBoss jBPM (versão 3.2 alpha 1): Faz uso de uma linguagem de workflow própria, o jBPM Process Definition Language (JPDL), e oferece suporte a Business Process Execution Language[33] (BPEL). Possui um bom editor visual integrado ao Eclipse e ao servidor de aplicações JBoss[34], o que oferece agilidade no momento a implantação/remoção de processos. Possui uma comunidade muito atuante. Oferece suporte à criação de formulários customizados que faça uso de JavaServer Faces.

- OBE – Open Business Engine (versão 1.0 RC1): A documentação deficiente e a instalação é bastante complexa. Não tem ferramenta própria para a criação de fluxos mas suporta o padrão XPDL.
- Enhydra Shark (versão 2.0 beta 1): Bem documentada, com comunidade atuante e um bom editor XPDL, no entanto faz uso de tecnologias pouco conhecidas, tais como o DODS (persistência objeto-relacional) e XMLC (tecnologia de interface com o usuário).
- WfMOpen versão 1.3.4: Bem documentada e aderente aos padrões XPDL e Xforms.
- Yawl versão beta 7: Apesar de fazer uso de uma linguagem de workflow própria, o Yet Another Workflow Language (YAWL). O Yawl impressiona pela qualidade da documentação e do embasamento técnico da pesquisa que o originou[35]. A comunidade é bastante atuante e a ferramenta está em franco amadurecimento.

Após a instalação e execução com sucessos dos exemplos que as acompanham, todas as ferramentas acima citadas ofereceram condições de implementação de fluxos variados de trabalho, inclusive os aqui pretendidos. Os critérios finais para a seleção da ferramenta a ser utilizada nessa prova de conceito foram a presença de comunidade ativa e o suporte de uma instituição reconhecida. Optou-se então pelo jBPM.

2.1 O jBPM

O jBPM é uma ferramenta bastante madura e muito conhecida entre os desenvolvedores Java. Inicialmente idealizada e desenvolvida por Tom Baeyens agora faz parte do pacote de middleware do servidor de aplicações J2EE JBoss, podendo ser executado de forma independente deste em ambientes bem mais simples tais como o Apache Tomcat ou em aplicativos Java não baseados em servlet, na forma de framework embutido. O jBPM é capaz de persistir as sessões de workflow em vários bancos de dados diferentes através do mecanismo de persistência objeto relacional Hibernate. A combinação dessas duas características fazem do jBPM uma boa opção tanto para organizações de maior porte que façam uso de ambientes computacionais complexos quanto para pequenas organizações que mantenham ambientes simplificados e totalmente baseados em software livre.

No aspecto do suporte, a comunidade é bastante atuante e todas as questões postadas no fórum são respondidas, seja por colaboradores oficiais JBoss seja por outros usuários. A documentação é abrangente e bem dividida, apesar de aparentemente estar um pouco defasada em relação à versão corrente.

Define-se um workflow no jBPM através de um arquivo no padrão XML, escrito segundo a definição da linguagem proprietária JPDL. O editor de fluxos distribuído com o jBPM é básico e deixa a desejar quando comparado a outras ferramentas semelhantes, tais como o Enhydra Jawe[36]. Além do JPDL, o jBPM suporta também o BPEL através de módulo de integração. A interface para execução e administração das instâncias dos processos é precária mas está em franco desenvolvimento.

A integração a outras ferramentas pode ser feita através de scripts embutidos na

definição do fluxo ou através de classes java que são referenciadas na definição e cuja funcionalidade é disparada de acordo com eventos que ocorrem durante a execução do mesmo. A API de acesso às funções do jBPM é bastante completa e bem documentada.

Apesar da versão corrente recomendada atualmente ser a 3.1.1, optou-se aqui por usar a versão 3.2 alpha 1 em função do suporte à customização de formulários oferecida nessa versão.

3 Implementação do processo de desenvolvimento de caso de uso

Inicialmente dois processos foram implementados usando o workflow automatizado como peça central: o de atendimento de chamados de suporte pelos usuários (processo de help desk) e o de desenvolvimento de casos de uso (processo de desenvolvimento de caso de uso), sendo que esses dois processos tiveram como base os processos adotados atualmente na Arcadian Tecnologia S/A[37].

Para esse trabalho optou-se por desprezar o processo de help desk e focar as avalições e possibilidades no processo de desenvolvimento de caso de uso, tanto por ser um processo que está está, de uma forma ou de outra, sempre presente em qualquer modelo de desenvolvimento quanto pelas possibilidades de automatização percebidas.

Ao se desenhar os processos no jBPM duas diretivas foram rigorosamente seguidas: não se alterou o fluxo funcionalmente e não se aumentou em nada a complexidade de uso para os participantes além do estritamente necessário para a notificação de aceitação e completude das tarefas no ambiente do jBPM.

3.1 Processo de desenvolvimento de caso de uso

Atualmente a Arcadian adota nos desenvolvimentos para a plataforma Java uma metodologia baseada em Model Driven Development[38] (MDA) e em um projeto típico pelo menos as seguintes fases de desenvolvimento são seguidas:

- 1) Levantamento de requisitos: levantamento, análise e especificação de requisitos.
- 2) Modelagem funcional básica: modelagem de alto nível dos casos de uso e geração do protótipo funcional.
- 3) Modelagem de persistência: definição das classes e relacionamentos entre elas.
- 4) Desenvolvimento dos casos de uso: modelagem detalhada dos casos de uso sua programação e testes.
- 5) Testes integrados: testes integrados da aplicação envolvendo todos os casos de uso.

A fase de desenvolvimento dos casos de uso contempla o seguinte fluxo:

- 1) o gerente de projetos delega a tarefa de modelagem para um analista ;
- 2) o analista modela o caso de uso de forma detalhada usando a ferramenta case MagicDraw[39], desenvolve um plano de testes para o caso de uso e notifica ao gerente o encerramento da tarefa, sendo que é comum que a modelagem seja liberada para a programação antes do desenvolvimento do plano de testes, e o desenvolvimento do plano de testes seja feito em paralelo à programação;

- 3) o gerente de projetos delega a tarefa de programação a um programador;
- 4) o programador usa a ferramenta MDA AndroMDA para gerar o código java básico no qual ele vai trabalhar, se o AndroMDA acusar erros de modelagem ele aciona o analista responsável pela modelagem para executar as correções e espera pela finalização das correções antes de continuar a programação;
- 5) após a geração de código bem sucedida o programador executa a programação das regras de negócio, faz testes básicos, coloca o executável disponível em ambiente de testes e notifica ao gerente o encerramento da tarefa;
- 6) o gerente delega a tarefa de testes a um testador, preferencialmente um outro analista;
- 7) o testador testa conforme o plano de testes e notifica ao gerente o encerramento da tarefa;
- 8) o gerente então dá a tarefa como terminada ou delega as correções ao analista ou ao programador, conforme seja o caso dos erros encontrados, mantendo o fluxo ativo.

A cada etapa o gerente é notificado do andamento pessoalmente ou através de e-mail e atualiza o cronograma do projeto de forma adequada.

Ao se desenhar esse fluxo no jBPM, percebeu-se a possibilidade de garantir o paralelismo do desenvolvimento dos casos de teste com a programação e a automatização das tarefas que envolviam a execução do AndroMDA chegando-se ao fluxo descrito na figura 1.

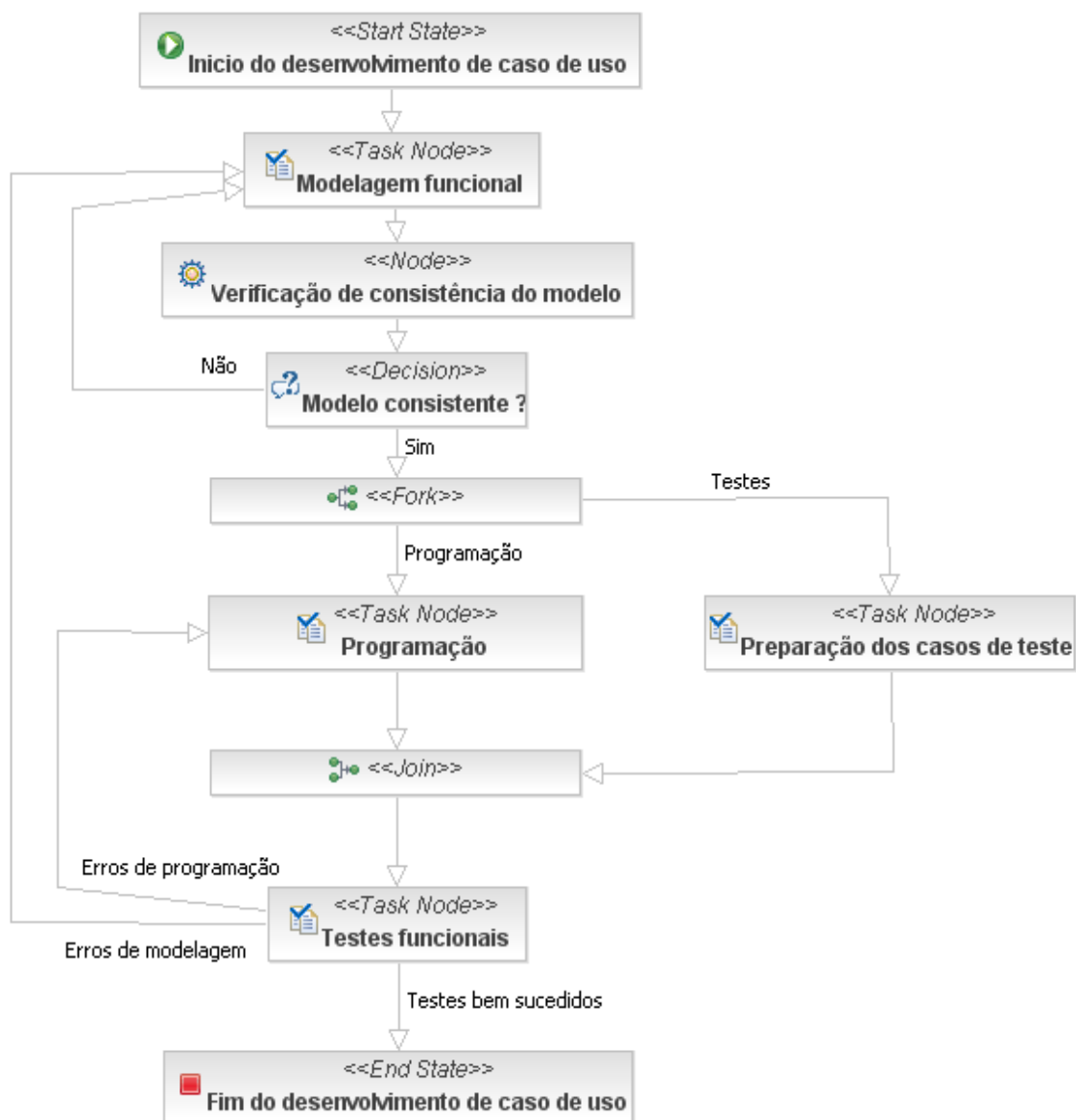


Figura 1 - Fluxo de desenvolvimento de caso de uso

3.2 A implementação desse fluxo no jBPM

Após o desenho do fluxo foram acrescentadas as variáveis mínimas necessárias para o seu bom andamento, tais como o nome do projeto no controle de versões e do caso de uso a ser desenvolvido, assim como os formulários necessários para a apresentação/edição dessas variáveis. Nenhuma dessas tarefas ofereceu maiores dificuldades uma vez que as variáveis podem ser inseridas no fluxo através do próprio editor visual e os formulários seguem o padrão XHTML e JavaServer Faces

Inicialmente três interfaces foram desenvolvidas para o jBPM: com o CVS[40], com o Maven e com os sistema de envio de e-mails do servidor de aplicações JBoss. Todas foram desenvolvidas em java, e suas funcionalidades são disparadas a partir da vinculação com eventos que ocorrem durante a execução do fluxo.

Foram criados os seguintes usuários participantes do fluxo: Gerente, Analista, Programador e Testador.

3.3 O passo a passo da execução do fluxo

Em um estado estacionário, as listas de tarefas de todos os participantes aparecem vazias, conforme a figura 2 que apresenta a lista de tarefas inicial do Analista.

Process Tasks You are logged in as Analista Log Out	Personal Task List					
	Task	Process	Duedate	Priority	Go	
	Group Task List					
	Candidates	Task	Process	Duedate	Priority	Go

Figura 2 - Lista de tarefas do analista

O fluxo de trabalho *Desenvolvimento de caso de uso* é ativado selecionando o link *Start It !* na lista de processos disponíveis, conforme figura 3, confirmando seu disparo após o preenchimento do formulário referente ao início do processo e pressionamento do botão *Salvar e fechar*, conforme figura 4.

Process Tasks Process Monitoring jBPM Administration You are logged in as Gerente Log Out	Personal Task List				
	Task	Process	Duedate	Priority	Go
	Group Task List				
	Candidates	Task	Process	Duedate	Priority
	Start new process execution				
	Process				Go
	Desenvolvimento de caso de uso				Start It!
	Abertura de chamado de Help Desk				Start It!
	websale				Start It!

Figura 3 - Lista de processos disponíveis para o gerente

Figura 4 - Formulário de início do processo

Uma vez que o gerente tenha iniciado o processo, a primeira ocorrência é a alocação da tarefa de modelagem funcional para o analista. Isso é notificado ao analista através da listagem da tarefa na sua lista de tarefas, conforme figura 5.

Personal Task List				
Task	Process	Due date	Priority	Go
Modelagem funcional	Desenvolvimento de caso de uso		3	Do it!

Figura 5 - Lista de tarefas do analista

Quando essa tarefa é aceita pelo analista, através do clique do mouse no link *Do it!*, ele tem acesso às variáveis do fluxo e deve executá-la de acordo seu padrão normal de trabalho. Ao final do trabalho de modelagem ele marca a tarefa como executada selecionando o botão *Salvar e fechar* do formulário da tarefa, conforme figura 6.

Figura 6 - Formulário da tarefa de modelagem funcional

Após a confirmação de execução da tarefa por parte do analista, o fluxo chega à primeira tarefa automatizada, a verificação da consistência do modelo.

Parte do processo de geração de artefatos do AndroMDA é a verificação da consistência do modelo. Essa etapa ocorre naturalmente antes da geração de código propriamente dita e impede o restante do geração caso não seja bem sucedida. Apesar dessa etapa poder ser executada pelo analista, tipicamente isso não ocorre para evitar que o analista tenha que lidar com a infra-estrutura necessária para a execução do AndroMDA. Sendo assim, no processo normal o programador gera o código a partir do modelo e notifica ao analista caso o modelo não esteja consistente. Esse tipo de situação naturalmente causa paradas e interrupções no processo e é um bom candidato para a automatização.

Elaborou-se essa automatização de modo que quando o fluxo chega à etapa de

verificação de consistência do modelo o AndroMDA é executado de forma transparente para os participantes e caso o modelo não esteja consistente o fluxo retorna à fase de modelagem funcional, notificando ao analista da falha ocorrida, conforme figura 7.

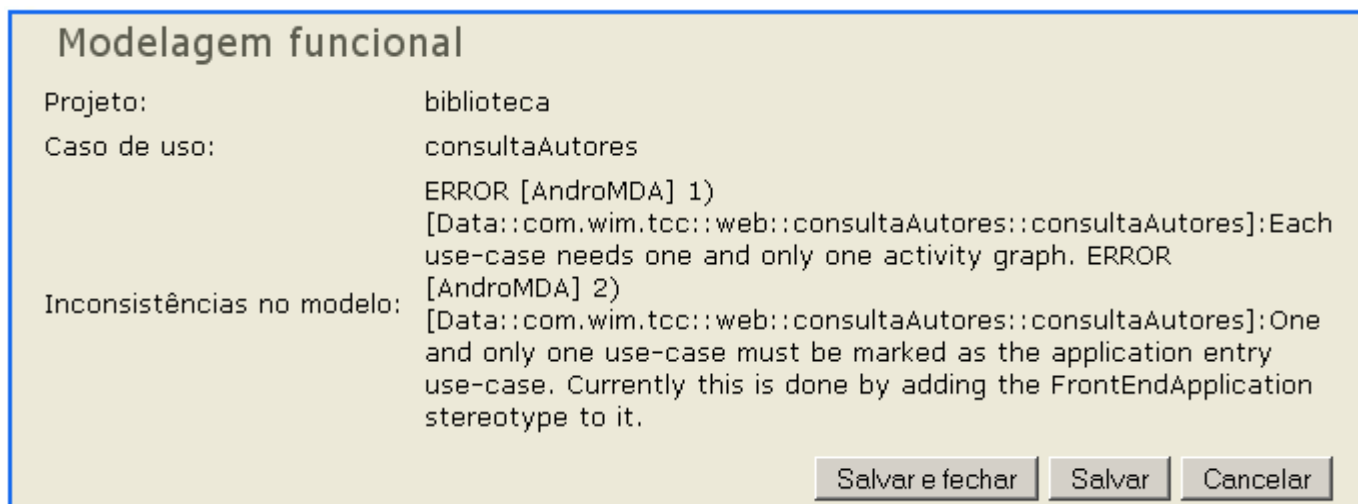


Figura 7 - Notificação de erro de consistência para o analista

A partir do momento em que o modelo está consistente e adequado à programação, a tarefa de preparação de casos de testes é alocada para o analista e a de programação é alocada para o programador simultaneamente, garantindo formalmente uma otimização que acontecia de modo informal.

Assim que as duas tarefas, programação e preparação dos casos de testes, forem encerradas, de modo análogo à de modelagem funcional, o testador recebe em sua lista de tarefas a executar a tarefa de testes funcionais. Diferentemente dos outros casos até o momento, ao encerrar a execução dos testes, o testador deve notificar optar entre três situações: *Erros de modelagem*, *Erros de programação* ou *Testes bem sucedidos*, conforme figura 8. Caso sejam localizadas falhas na modelagem o fluxo volta para a modelagem funcional, em caso de falhas de programação o fluxo volta para a programação e caso o teste seja bem sucedido o fluxo é finalizado.

Testes funcionais

Projeto: biblioteca

Caso de uso: consultaAutores

Erros de modelagem:

Erros de programação:

Erros de modelagem Erros de programação Testes bem sucedidos

Figura 8 - Formulário de execução de testes funcionais

Os ganhos percebidos nessa primeira implementação frente ao processo atualmente estabelecido se resumiram à automatização do processo de verificação de consistência do modelo e formalização da concorrências entre as tarefas *Programação* e *Preparação dos casos de testes*. Poderia-se também observar a melhoria de visibilidade no andamento das tarefas por parte do gerente, no entanto esse não é um ganho significativo uma vez que qualquer ambiente de desenvolvimento com um mínimo de organização já deve contemplar esse quesito de forma eficiente.

4 Implementação de melhorias no processo

Feita uma análise do processo implementado, algumas áreas de melhoria associadas ao uso de uma ferramenta de workflow foram identificadas e implementadas, conforme segue. Foi também implementada uma nova interface com a ferramenta de medições de código Java JxRef.

4.1 Vinculação de documentos ao processo e à tarefa

Acrescentou-se ao pacote do processo um conjunto de arquivos que simulam a presença de documentos definindo as melhores práticas e padrões para cada etapa do processo, assim como links para esses documentos e para o repositório de documentos específicos do projeto, conforme figura 9.

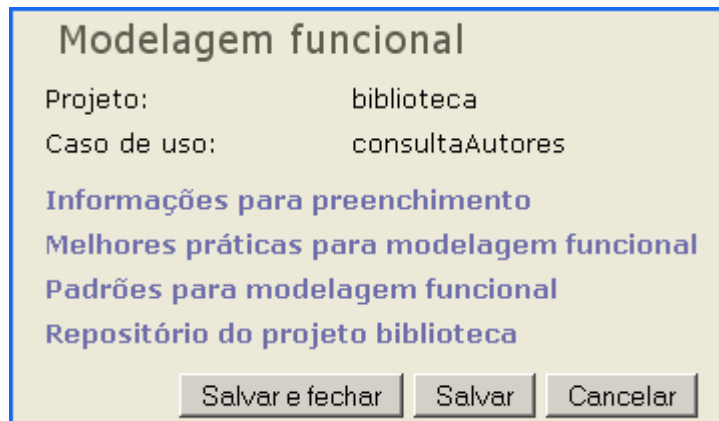


Figura 9 - Links para documentos relevantes e repositório

A vinculação de documentos ao processo e seu fácil acesso durante a execução das tarefas exemplifica o papel de repositório que a ferramenta de workflow pode representar. No exemplo em estudo optou-se por exemplificar a vinculação com os artefatos gerados através de um link html com o repositório de documentos, no entanto percebeu-se que nada impede que os artefatos sejam efetivamente gravados juntamente com a instância do processo que os gerou na forma de variáveis do processo, lembrando que essas variáveis podem ser objetos de qualquer tipo, inclusive anexos em quaisquer formatos. A priori os próprios erros gerados durante as etapas de verificação de consistência do modelo e de testes funcionais e o relatório de qualidade da programação (será visto a frente), podem ser considerados artefatos da instância do processo e estão associados a essa instância de forma permanente.

4.2 Extração de medições e automatização de controles de qualidade

Foram feitas duas modificações relacionadas à coleta e análise de medições de forma automatizada. Inicialmente incluiu-se uma etapa de medições logo após a etapa de programação com o objetivo de executar o analisador JxRef no código do caso de uso programado, gerando e anexando um relatório à instância do fluxo conforme figura 10. Isso possibilitou que o testador fizesse a análise do relatório e eventualmente recusasse o código programado.

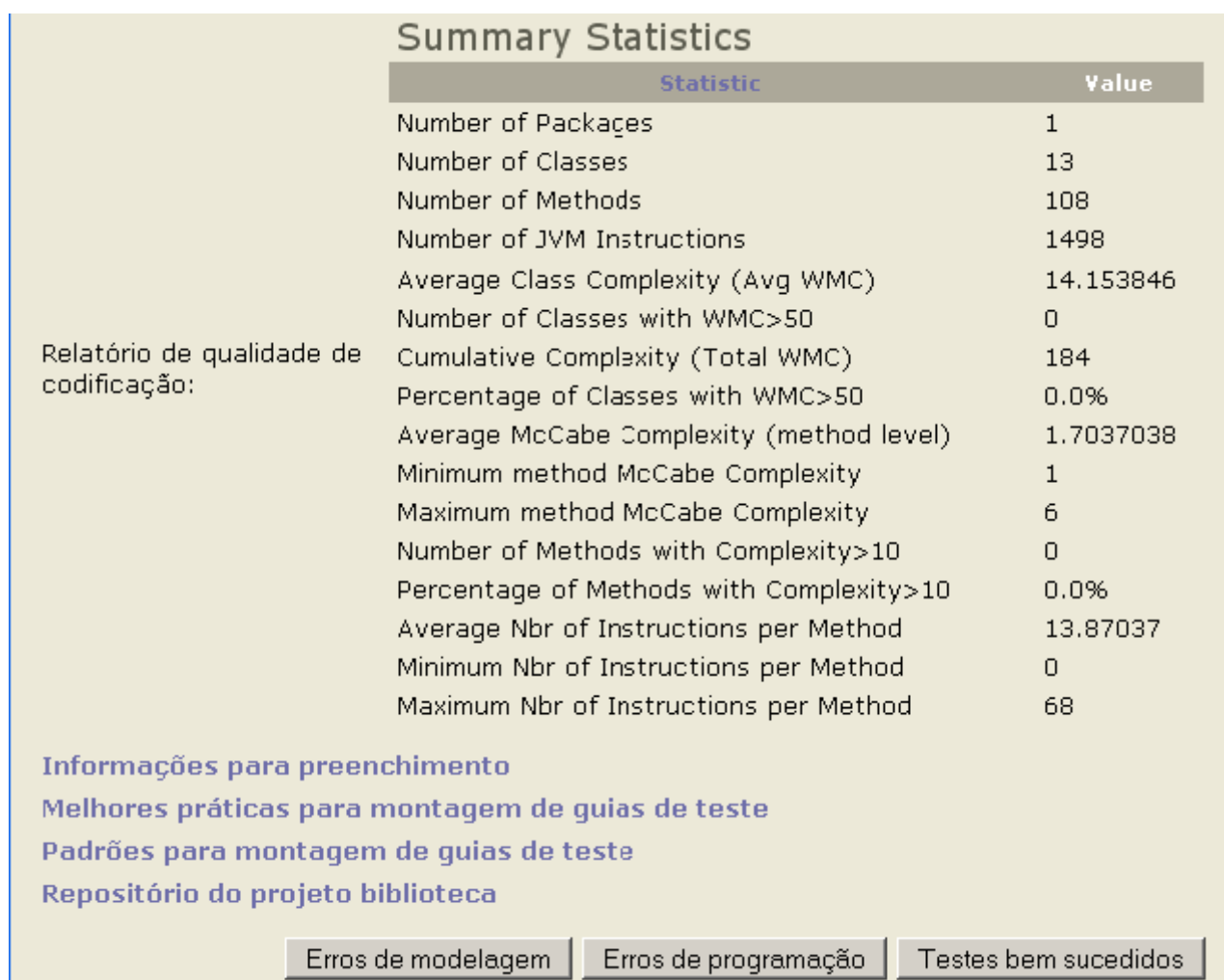


Figura 10 - Relatório de complexidade do caso de uso junto à tarefa de testes

Em um segundo momento transferiu-se parte dessa responsabilidade de análise e recusa do código programado para uma tarefa automatizada de modo que determinadas medições que não estejam dentro de normas pré-definidas causem a recusa automática, fazendo com que o fluxo volte à etapa de programação descrevendo para o programador o motivo do retorno conforme figura 11. Para a implementação no exemplo considerou-se impeditivo para a continuidade do fluxo a existência de métodos cujo índice de complexidade de McCabe[41] fosse superior a 10.

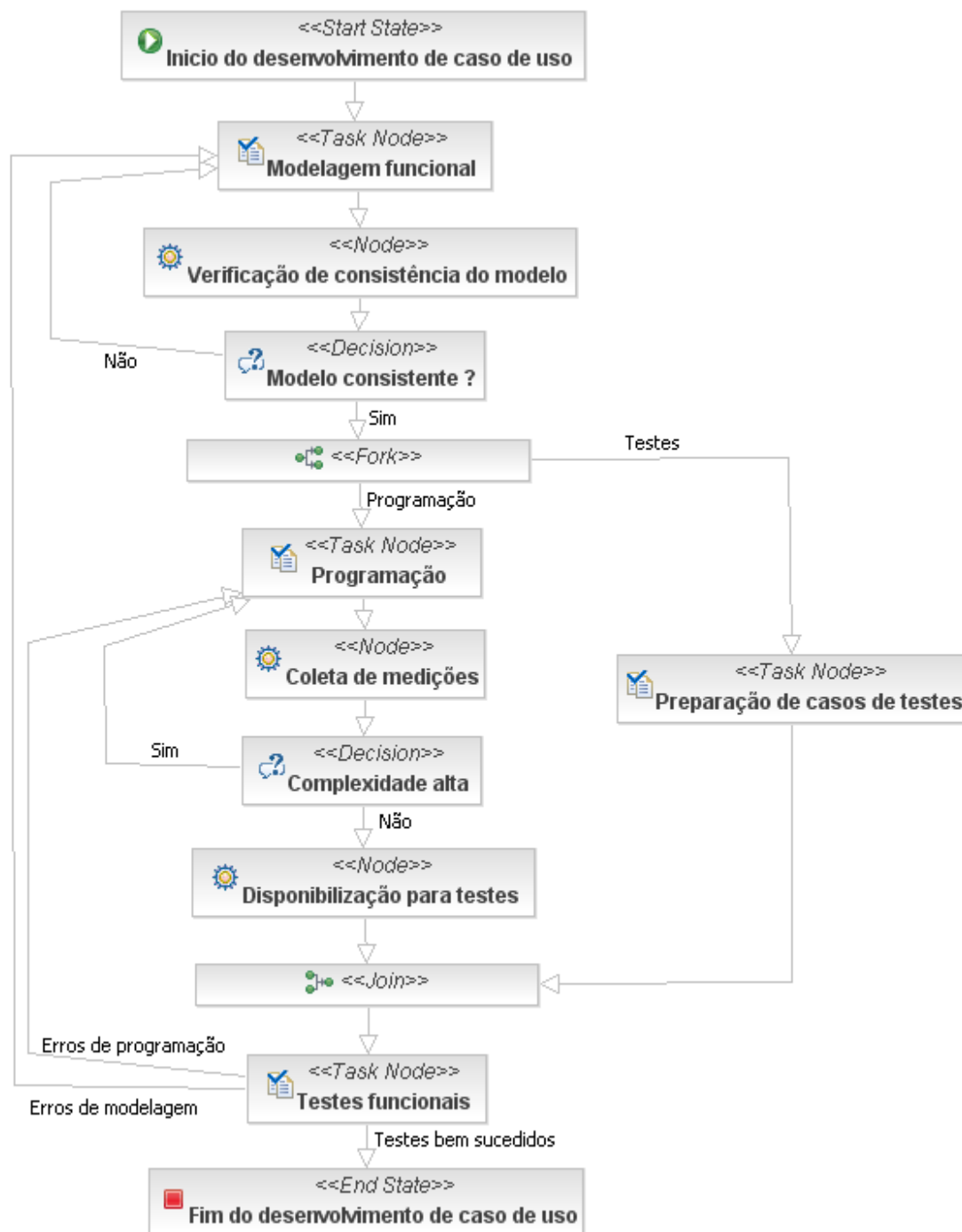


Figura 13 - Fluxo final do desenvolvimento de caso de uso

4.5 Dificuldades encontradas na implementação dos processos

Uma vez que optou-se por usar uma versão do jBPM que ainda está em desenvolvimento, a 3.2 alpha 1, já se esperava que uma série de bugs e anomalias fosse encontradas, como realmente aconteceu. No entanto todos os defeitos encontrados foram de fácil contorno ou solução uma vez que a grande maioria dizia respeito a aspectos da interface com o usuário e o núcleo de processamento e persistência se mostrou totalmente estável, conforme cabe a uma ferramenta madura.

Em relação às interfaces previstas, à exceção do JxRef, todas as dificuldades foram relativas ao conhecimento das ferramentas envolvidas sendo sanadas com consultas aos sites dos desenvolvedores e outros de informações, tutoriais e fóruns.

O JxRef foi desenvolvido para ser uma tarefa do Apache Ant e não estava preparado para ser executado em um ambiente de acesso restrito ao sistema operacional como é o caso de servlets que são executados no JBoss. Várias interferências foram feitas em seu código fonte para que seu funcionamento acontecesse de forma adequada e estável.

5 Conclusões e proposta de trabalhos futuros

Conclui-se, após as observações aqui relatadas, que o uso de ferramentas de workflow em processos de engenharia de software pode trazer ganhos consideráveis quando a ferramenta está bem integrada ao restante do ambiente. Além das funções de repositório de processos, monitoramento da instância do processo, automatização de integrações e geração de métricas inicialmente previstos, observou-se que a ferramenta de workflow pode ser um elemento ativo na tomada de decisões que garantam determinados parâmetros de qualidade tanto a partir de medições genéricas quanto a partir de medições específicas do processo em questão.

5.1 Ferramentas de workflow em organizações buscando certificações/avaliações

Merece destaque a possível utilização da ferramenta de workflow para a extração de evidências objetivas em organizações que estejam em busca de certificações ISO ou avaliações CMMI ou MPS-BR, a partir do armazenamento de artefatos junto às instâncias do processo.

Adicionalmente percebe-se que organizações que estejam buscando níveis altos na avaliação CMMI ou MPS-BR podem fazer uso das informações geradas automaticamente pelas ferramentas nas análises necessárias nos ambientes de melhoria contínua de processos.

5.2 Trabalhos futuros

Um aspecto que merece ser melhor estudado é o uso da ferramenta de workflow como garantia de aderência metodológica em ambientes informais, ou seja: determinar que influência a ferramenta de workflow pode ter na garantia da adoção e execução dos preceitos metodológicos determinados para a organização, em particular no que concerne às metodologias ditas ágeis, de modo a agregar a burocracia necessária sem perda da agilidade desejada.

Finalmente, a principal e mais imediata proposta de trabalho futuro que se

percebe é a ampliação do presente trabalho no aspecto prático, através da incorporação de processos que compreendam uma parte significativa de um projeto de software completo e sua efetiva utilização em ambiente de desenvolvimento.

Referências

- [1] Baeyens, Tom (2004) "The State Of Workflow". Disponível em: <<http://www.jboss.com/products/jbpm/stateofworkflow>>
- [2] Becker, Jörg;Muehlen, Michael; Gille Marc (2003) "Workflow Application Architecture: classification and Characteristics of Workflow-based Information Systems", Workflow Handbook 2002, MIT Press
- [3] MALONE, Thomas W.; Crowston, Kevin; Herman, George A. (2003) "Organizing Business Knowledge"
- [4] Salviano, Clênio F. (2004) "Melhoria e Avaliação de Processo com ISO/IEC 15504 (SPICE) e CMMI", UFLA
- [5] Reis, Glauco (2006) "O que é uma solução BPMS". Disponível em: <<http://www.portalbpm.com.br/servlet/leartigo?qual=/WEB-INF/artigos/decisao/file2.pdf>>
- [6] Acesso em: <<http://maven.apache.org/>>
- [7] Acesso em: <<http://www.jxref.org/>>
- [8] Acesso em: <<http://www.eclipse.org/>>
- [9] Acesso em: <<http://sourceforge.net/>>
- [10] Acesso em: <<http://www.objectweb.org/index.html>>
- [11] (Baeyens, Tom) "". Disponível em: <>
- [12] Acesso em: <<http://bonita.objectweb.org/>>
- [13] Acesso em: <<http://www.wfmc.org/>>
- [14] Acesso em: <<http://www.loria.fr/equipes/ecoo/index.html>>
- [15] Acesso em: <<http://www.jboss.com/products/jbpm>>
- [16] Acesso em: <<http://nautica.sourceforge.jp/index-e.html>>
- [17] Acesso em: <<http://www.wfmc.org/standards/XPDL.htm>>
- [18] Acesso em: <<http://obe.sourceforge.net/>>
- [19] Acesso em: <<http://web.openwfe.org/display/openwfe/Home>>
- [20] Acesso em: <<http://www.opensymphony.com/osworkflow/>>
- [21] Acesso em: <<http://sourceforge.net/projects/powerstone>>
- [22] Acesso em: <<http://springframework.org>>
- [23] Acesso em: <<http://www.hibernate.org>>
- [24] Acesso em: <<http://wf.runa.ru/English/About/About.html>>

- [25] Acesso em: <<http://forge.objectweb.org/projects/shark>>
- [26] Acesso em: <<http://www.omg.org/>>
- [27] Acesso em: <<http://swamp.sourceforge.net/>>
- [28] Acesso em: <<http://wfmopen.sourceforge.net/>>
- [29] Acesso em: <<http://java.sun.com/javaee/>>
- [30] Acesso em: <<http://www.yawl.fit.qut.edu.au/about/>>
- [31] Acesso em: <<http://www.w3.org/MarkUp/Forms/>>
- [32] Acesso em: <<http://forge.objectweb.org/projects/exoplatform>>
- [33] Acesso em: <<http://en.wikipedia.org/wiki/BPEL>>
- [34] Acesso em: <<http://www.jboss.org>>
- [35] Aalst W.M.P.r, Hofstede A.H.M. () "YAWL: Yet Another Workflow Language". Disponível em: <<http://www.yawl.fit.qut.edu.au/yawldocs/yawlrevtech.pdf>>
- [36] Acesso em: <<http://www.enhydra.org/workflow/jawe/index.html>>
- [37] Acesso em: <<http://www.arcadian.com.br>>
- [38] Acesso em: <<http://www.omg.org/mda/>>
- [39] Acesso em: <<http://www.magicdraw.com/>>
- [40] Acesso em: <<http://www.nongnu.org/cvs/>>
- [41] BRUSAMOLIN, Valério () "Manutenibilidade de Software". Disponível em: <http://www.revdigonline.com/artigos_download/art_10.pdf>