INSTITUTO MILITAR DE ENGENHARIA

FÁBIO RACHID DA ROCHA

INSTANCIAÇÃO E EXECUÇÃO DE MODELOS DE PROCESSO DE SOFTWARE NO ECLIPSE PROCESS FRAMEWORK

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Ricardo Choren Noya – D. Sc.

Rio de Janeiro

2007

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

c2007

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

R672 Rocha, Fábio Rachid da
Instanciação e Execução de Modelos de Processo de Software no Eclipse Process Framework / Fábio Rachid da Rocha. – Rio de Janeiro: Instituto Militar de Engenharia, 2007.
125 p.: il.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2007.

1. Engenharia de Software. 2. Processo de Software. 3. Software Livre. 4. Integração. I. Título. II. Instituto Militar de Engenharia

CDD 005.1

INSTITUTO MILITAR DE ENGENHARIA

FÁBIO RACHID DA ROCHA

INSTANCIAÇÃO E EXECUÇÃO DE MODELOS DE PROCESSO DE SOFTWARE NO ECLIPSE PROCESS FRAMEWORK

Dissertação de Mestrado apresentada ao Curso de Mestrado Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Ricardo Choren Noya - D. Sc.

_____

Prof. Ricardo Choren Noya - D. Sc. do IME - Presidente

_____

Prof. Cláudio Gomes de Mello - D.Sc. do IME

_____

Prof. Márcio de Oliveira Barros - D.Sc. da UNIRIO

Rio de Janeiro

2007

Dedico este trabalho a minha família e meus amigos.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus pela força e aos meus pais, pelo incentivo e apoio dado durante todo este tempo.

Ao meu orientador Prof. Ricardo Choren, pela dedicação, orientação e ajuda.

Ao amigo Prof. Cícero Garcez, pela colaboração, paciência e dedicação.

Agradeço a todos os amigos do IME, que juntos conseguimos atingir nossos objetivos.

A Capes, pelo apoio financeiro concedido no início do curso.

A todos os funcionários e professores do departamento de Engenharia da Computação e Telemática do IME (SE/8), meu agradecimento e admiração.

Fábio Rachid da Rocha

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ADS	Ambiente de Desenvolvimento de Software
AIPS	Ambiente Integrado para Processos de Software
CASE	Computer-aided Software Engineering
EPF	Eclipse Process Framework
IDE	Integrated Development Environment
JSP	Java Server Page
MDS	Método de Desenvolvimento de Sistemas
OMG	Object Management Group
OPENUP	Open Unified Process
PRODERJ	Centro de Tecnologia da Informação e Comunicação do Estado do Rio de Janeiro
PSEE	Process-Centered Software Engineering Environments
RUP	Rational Unified Process
SADT	Structured Analysis and Design Technique
SPEM	Software Process Engineering Metamodel
SWT	Standard Widget Toolkit
UEM	Universidade Estadual de Maringá
UMA	Unified Method Architecture
UML	Unified Modeling Language
URL	Uniform Resource Locator
WBS	Work Breakdown Structure
XML	Extensible Markup Language

RESUMO

Com o objetivo de aumentar a qualidade dos produtos de software, é importante definir e controlar o processo de desenvolvimento. Um Ambiente de Desenvolvimento de Software orientado ao processo (ADS) deve contemplar as fases de modelagem, instanciação e execução de forma integrada. Assim, um ADS fornece um mecanismo de controle que possibilite ao gerente estar ciente do estado e das ocorrências do projeto, além de contribuir para a atividade contínua de melhoria de processo.

Este trabalho pesquisa uma proposta de modelo e mecanismos que contribuam para o controle do processo de desenvolvimento de software, envolvendo a definição, instanciação e execução integrada de processos. O ponto inicial da proposta é a ferramenta de modelagem *Eclipse Process Framework* (EPF, 2006), que permite a definição de modelos de desenvolvimento. Foram criados mecanismos para a instanciação e execução desse modelo, que trabalham de forma integrada à ferramenta de modelagem e propõem uma maneira de controlar o processo que ofereça o conjunto de informações consideradas relevantes para os atores e gerentes de projeto.

O mecanismo de instanciação foi implementado como um *plugin* para o Eclipse, que utiliza como entrada o produto gerado pela ferramenta de modelagem e possibilita a definição de um conjunto de informações que transformem o modelo abstrato de um processo de software em um modelo executável. Neste conjunto de informações estão os prazos para as tarefas e as ferramentas que devem ser utilizadas durante sua realização, além da definição de atores e associação dos papéis aos quais os atores fazem parte.

O mecanismo de execução foi implementado como um sistema Web que utiliza como entrada o produto gerado pela ferramenta de instanciação. O mecanismo oferece informações aos atores, os guiando ao longo do projeto. Deve ser utilizado como repositório de documentos, centralizando o arquivamento dos mesmos e facilitando o acesso por parte de qualquer um dos participantes do projeto. Permite o registro de ocorrências para cada tarefa e modificações dinâmicas ao longo da execução do processo.

ABSTRACT

When goal is to increase software quality, it is essential to define and control development process. The Process-Centered Software Engineering Environment (PSEE) must contemplate process modeling, instancing and enactment phases integrated. This way, a PSEE offers a control mechanism that allows the manager to be informed about project status and issues, as well as to contribute with evolutive process enhancement.

This research evaluates a model and mechanisms proposal that contributes to software developing process control, which involves process definition, instancing and enactment. The start point of the proposal is the Eclipse Process Framework modeling tool (EPF, 2006), which allows the definition of development models. Mechanisms, which were created to instance and execute this model, work integrated to the modeling tool and offer the opportunity to control the process in such a way that all information that actors and project managers consider important are being held.

The instantiation mechanism was implemented as an Eclipse plugin that uses the product generated by the modeling tool as an input, allowing the definition of an information pool that will turn the abstract model of a software process into an executable model. This information pool displays to do's deadlines and tools to be used during their accomplishment, as well as actor's definitions and their tasks.

The process enactment mechanism was implemented as a web application that uses the product generated by the instantiation plug-in as an input. This mechanism offers information to the process actors, guiding them throughout the project. It is supposed to be used as documents repository, making them easier to file and access by anyone on the project. It also allows registering issues for each task and dynamic modifications throughout the process enactment.

1. INTRODUÇÃO

Processo é a realização sistemática de uma ou mais atividades para a criação de um produto (OSTERWEIL, 1987). O processo de software é o conjunto de todas as atividades relacionadas ao desenvolvimento, controle, validação e manutenção de um software operacional, compreendendo desde a definição de requisitos e elaboração de protótipos, até a validação do software e o controle de qualidade (PRESSMAN, 1997).

Rabelo (2001) afirma que processos de software são complexos e envolvem um grande número de pessoas com tarefas especializadas, equipes multidisciplinares e cronogramas de longo prazo. Além disso, o processo de software envolve atividades técnicas e gerenciais, compreendendo métodos, ferramentas de desenvolvimento e diversos recursos, formando uma grande rede de inter-relacionamentos e interdependências. Toda esta complexidade acentua a dificuldade de percepção do processo como um todo por parte dos profissionais envolvidos, os quais necessitam possuir visões (ou subconjuntos de informações) apresentadas de forma adequada e orientadas ao seu papel no processo (por exemplo, desenvolvedor, gerente, dentre outros).

Por isso, um processo de desenvolvimento de software deve possuir uma descrição explícita, clara, detalhada e formal a fim de melhorar a qualidade do software produzido. O crescente interesse pelo processo de software demonstrado por parte da indústria, se deve ao fato de que um processo bem definido permite um monitoramento e controle do desenvolvimento de software e, dessa forma leva à redução de custos de produção, bem como à melhoria da qualidade e integridade do software produzido (GIMENES, 1994).

Em relação ao processo de software, existem três principais ações que podem ser realizadas: definir ou modelar, avaliar e melhorar. A modelagem de um processo de software deve explicitar o conjunto de atividades que devem ser realizadas pela organização, indicando como e quem deve realizá-las, para gerar o produto final. A avaliação envolve analisar as atividades realizadas em uma organização, com o objetivo de melhorar a produção do software. A melhoria do processo estuda a maneira de melhorar as práticas de desenvolvimento de software em uma organização, uma vez que a avaliação do processo esclareceu o seu estado (ACUÑA, 2000). Então, ter o processo definido possibilita a avaliação e posterior melhoria do processo.

A Tecnologia de Processos de Software propõe o desenvolvimento e adoção de Ambientes de Desenvolvimento de Software Orientados ao Processo para automatizar a

gerência dos processos (GIMENES, 1994). Esta tecnologia traz benefícios, como por exemplo: melhor comunicação entre as pessoas envolvidas e a consistência do que está sendo feito; realização de algumas ações automáticas, liberando os seus usuários de tarefas repetitivas; fornecimento de informações sobre o andamento do processo quando necessário; possibilidade de reutilização de processo de software e a coleta automática de métricas (importantes para o controle e aperfeiçoamento de processos) (REIS, 2003).

1.1. DESCRIÇÃO DO PROBLEMA

Todo software é produzido através de algum processo. Entretanto, na maioria das vezes este processo é incoerente e está implícito, levando à falta de previsibilidade, falta de capacidade de repetição dos passos e falta de uma base que permita aperfeiçoamento (REIS, 2003). Tornando este processo explícito muitos destes problemas podem ser solucionados, pois a partir de um modelo de processo é possível monitorar, compreender e evoluir o mesmo. Em um processo de desenvolvimento existe a interação entre pessoas e a interação de pessoas com ferramentas, o que torna o seu andamento variável e imprevisível. Esta característica aumenta a complexidade do processo e da gerência da sua execução, estendendo pelas atividades a serem coordenadas.

Um dos grandes obstáculos para o surgimento de uma indústria de software de porte consiste na disponibilização de ferramental tecnológico que atue decisivamente na gerência dos processos conduzidos nas organizações, controlando prazos, gerenciando a alocação de recursos e mantendo controle sobre os produtos resultantes, trazendo para esta área muito da disciplina que influencia o desenvolvimento bem sucedido de indústrias de outras áreas (MOITRA, 2001). Tais ferramentas, quando disponíveis, envolvem custos que ultrapassam em muitas vezes a capacidade de investimento das organizações.

Um Ambiente de Desenvolvimento de Software (ADS) deve proporcionar um alto nível de integração entre suas ferramentas e ser capaz de executar um modelo de processo de software, através da coordenação dos desenvolvedores na execução de suas tarefas, permitindo coleta de métricas, execução automática de algumas atividades e mudança do processo durante sua execução (LIMA, 1998).

A modelagem e a execução de processos de software é suportada por ADS orientados ao processo (em inglês, *Process-Centered Software Engineering Environments - PSEEs*), que são ambientes de desenvolvimento de software orientados ao processo. Nestes ambientes, um

modelo de processo é interpretado por um mecanismo de execução, o qual interage com os desenvolvedores e com as ferramentas do ambiente a fim de controlar o processo de software (LIMA, 1998). A utilização de ADS orientados ao processo traz benefícios como: melhor comunicação entre as pessoas envolvidas no desenvolvimento de software, realização de algumas atividades de forma automática, coleta de métricas, suporte à reutilização de modelos de processo e disponibilidade de informações sobre o andamento do processo de software (REIS, 2003). Cada PSEE é caracterizado pela sua linguagem de modelagem, que deve suportar a descrição dos vários conceitos que caracterizam o processo de desenvolvimento de software, como (CUGOLA, 1998): atividades, artefatos de software, papéis, agentes humanos, recursos e ferramentas.

Existem ambientes de desenvolvimento de software centrados em processos que trabalham de forma integrada fornecendo mecanismos para definição, instanciação e acompanhamento de processos de software como o ExPSEE (FANTINATO, 1999) e o WebAPSEE (FRANÇA, 2006). Porém há falta de um mecanismo de acompanhamento que possibilite aos atores registrar suas ocorrências, que são informações importantes ao gerente do projeto, para que o mesmo seja capaz de compreender o motivo pelo qual o processo se encontra em determinado estado de execução.

1.2. OBJETIVO

Levando em consideração os problemas relatados na seção anterior, o objetivo deste trabalho é oferecer um ambiente integrado para a definição, instanciação e controle de processos de desenvolvimento de software. Este ambiente integrado deve suportar as três fases, garantindo que o produto gerado por uma fase sirva de entrada para a fase seguinte. Esta transição deve ocorrer de forma natural, sem a necessidade de adaptação por parte do usuário.

Este ambiente deve fazer uso de recursos de código aberto, tendo em vista o forte incentivo dado pelo governo, tanto no âmbito Federal quanto Estadual, em suas instituições e autarquias em relação à distribuição e utilização de ferramentas livres, disponíveis atualmente, permitindo automatizar o acompanhamento de todo o processo. A partir da modelagem, utilizada para explicitar o processo, serão criadas instâncias, que permitirão um melhor acompanhamento e gerência para cada projeto de desenvolvimento onde o mesmo for

aplicado. Com isso, ao se executar a instância do processo será possível guiar os atores ao longo de todo o projeto, além de oferecer ao gerente uma visão geral do andamento.

Para facilitar o acesso por parte de todos os componentes da equipe, a fase de execução deverá ser disponibilizada na Web. Com o objetivo de fornecer uma visão clara do andamento das tarefas, deve possibilitar aos atores registrarem a história do processo de forma detalhada o suficiente, para que mesmo geograficamente distante, o gerente possa acompanhar as ocorrências do projeto e assim melhorar ou corrigir o processo a fim de atingir o seu objetivo.

Alguns trabalhos existentes, como o ExPSEE e o WebAPSEE, atendem alguns desses requisitos, mas não todos. O ExPSEE é uma aplicação desenvolvida com arquitetura cliente servidor, que utiliza uma linguagem visual própria para definição do processo. Durante a execução, permite aos atores apenas alterarem o estado das tarefas, sem um maior detalhamento sobre o seu andamento. O WebAPSEE, apesar de ser um ambiente todo desenvolvido com arquitetura Web, também utiliza uma linguagem visual própria para definição do processo (o WebAPSEE-PML) e também permite apenas a alteração dos estados das tarefas por parte dos atores.

1.3. CONTRIBUIÇÕES

Ao atingir os objetivos anunciados na seção anterior, espera-se que haja contribuições na área de definição, controle e acompanhamento de processos de software. A lista de contribuições esperadas é a seguinte:

- Prover, de forma integrada um mecanismo de modelagem, instanciação e execução de processos de software, criando um ambiente integrado para a gerência de processos de desenvolvimento.
- Oferecer um mecanismo de controle de execução de processos que disponibilize aos atores do processo informações detalhadas de suas tarefas.
- Desenvolver uma ferramenta que permita a gestão de processos de desenvolvimento.
- Desenvolver as ferramentas como software livre, para que estes possam sofrer evoluções por parte dos usuários e participantes da comunidade.

1.4. ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação apresenta todo o estudo feito em relação ao tema, apresentando uma descrição do domínio do problema, aspectos identificados como sendo importantes para processos de software e o conhecimento adquirido através dos resultados e das conclusões obtidas a partir da instanciação e execução dos processos modelados pela ferramenta *Eclipse Process Framework*.

No Capítulo 2, é dada uma maior fundamentação em relação aos assuntos pesquisados. É apresentada uma seção relacionada à processos de software, onde é discutido a modelagem, instanciação e execução de processos além do *framework* de modelagem de processo *Eclipse Process Framework* (EPF), estudado e utilizado na validação dos tópicos aprendidos e como prova de conceito dos trabalhos desenvolvidos. Por último será apresentada uma seção descrevendo a execução de processo de software e o funcionamento de mecanismos que realizam tal execução.

No capítulo 3, são apresentados os trabalhos utilizados como fonte de referência para os estudos acerca de ambientes integrados de desenvolvimento de software centrados em processo. Ao fim do capítulo, uma seção apresenta considerações sobre os trabalhos relacionados.

No capítulo 4, é apresentado o Ambiente Integrado para Processos de Software (AIPS) como proposta de um ambiente integrado para modelagem, instanciação e execução de processos de software. O mecanismo responsável por cada uma destas fases é apresentado, e um processo exemplo é utilizado com o objetivo de ilustrar as transformações sofridas pelo modelo ao longo de todo o processo.

No capítulo 5, é apresentado o estudo de caso, como mecanismo para a prova de conceito, aplicado sobre a ferramenta EPF, mostrando a execução da instância do processo desenvolvido e utilizado pelo PRODERJ e o OpenUP/Basic.

No Capítulo 6, é relatado todo o trabalho de conclusão, obtido através dos resultados alcançados, a partir de lições aprendidas pelos experimentos feitos. Além disso, são propostos neste capítulo alguns trabalhos futuros, visando obter maior maturidade para a pesquisa feita.

2. FUNDAMENTAÇÃO TEÓRICA

Este Capítulo trata dos fundamentos conceituais sobre os quais o trabalho apresentado nesta dissertação foi construído. Desta forma são apresentados a seguir os tópicos que orientam no embasamento para a formalização do estudo aqui aplicado.

2.1. PROCESSOS DE SOFTWARE

O processo de software é o conjunto de todas as atividades relacionadas ao desenvolvimento, controle, validação e manutenção de um software operacional, compreendendo desde a definição de requisitos e elaboração de protótipos, até a validação do software e o controle de qualidade (PRESSMAN, 1997).

Processos de software devem acomodar atividades desenvolvidas por pessoas. Essas atividades são caracterizadas pela criatividade, informalidade, julgamento e por atividades que não tem qualquer tipo de apoio automatizado, como em reuniões e negociações. Em contrapartida a esses fatores, estão os objetivos das linguagens de processo, que procuram facilitar a definição dos processos de software de alta qualidade, sob a ótica da Engenharia de Software (SUTTON e OSTERWEIL, 1995). Tendo em vista disso, qualquer representação de um determinado processo é um modelo deste processo (SILVA, 2001). Diversos tipos de informação devem ser incluídos no modelo a fim de indicar quem, quando, onde, como e por que cada um dos passos é realizado (LONCHAMP, 1993).

Uma equipe de desenvolvimento que trabalha sem um processo bem definido acaba funcionando de maneira *ad hoc*, o que compromete o seu sucesso, criando uma situação insustentável. Por outro lado, organizações maduras empregando um processo bem definido podem desenvolver sistemas complexos de maneira consistente e previsível, independente de quem o produziu (KRUCHTEN, 1999) (ROCHA, 2001).

Em vista disso, a vantagem de se ter um processo de software bem definido é o oferecimento de uma forma de analisar e amadurecer tal processo, através da sua descrição, a qual permite que o processo seja analisado, compreendido e eventualmente evoluído. Esse amadurecimento do processo é uma das características que influenciam na qualidade do produto gerado.

2.1.1. MODELAGEM

O modelo de um processo de software é a representação formal dos elementos envolvidos neste processo onde cada elemento pode ser executado por pessoas ou máquinas. As descrições de ciclo de vida representam, em sua maioria, um modelo abstrato de desenvolvimento de software, não fornecendo informações sobre como integrar os diversos passos do processo que são realizados pelas pessoas envolvidas. Um dos objetivos da modelagem de processo é detalhar o suficiente estas descrições para orientar a execução do processo (GIMENES, 1994).

Segundo (ACUÑA, 2000), podem-se modelar diferentes elementos de um processo como, por exemplo: atividades, produtos (artefatos), atores e papéis.

- **Atividade**: é o estágio que produz mudanças visíveis de estado no produto de software. Pode ter uma entrada, uma saída e alguns resultados intermediários, normalmente chamados de produtos. A atividade inclui e implementa procedimentos, regras, políticas e objetivos para gerar e modificar um conjunto de determinados artefatos. As atividades podem ser divididas em atividades mais simples; isto é, existem atividades simples e compostas.
- **Artefato ou Produto**: é o produto ou subproduto de um processo ou atividade. Um artefato produzido por um processo pode ser usado como fonte de material para o mesmo ou algum outro processo produzir outro artefato. Um artefato ou produto de software é desenvolvido ou mantido por um processo. Os (sub) produtos podem ser criados, acessados ou modificados durante o processo. Um conjunto de artefatos de software a ser entregue a um usuário é chamado produto de software.
- **Ator**: é a entidade que executa o processo. Atores podem ser divididos em dois grupos: atores humanos ou atores de sistema. Um ator é caracterizado pela descrição do seu papel e pode desempenhar diversos papéis.
- **Papel**: define o conjunto de atores ou de responsabilidades, deveres e conhecimento necessário para se executar uma atividade.

Modelar o processo de software permite entender e apoiar o desenvolvimento de sistemas de software. A modelagem de processos de software não se resume apenas na escrita de programas que automatizem o processo de desenvolvimento de software, nem em descrever

tudo o que os envolvidos no processo devam fazer (REIS, 1999). Enquanto os programas de computador são escritos para definir o comportamento de uma máquina determinística, os programas de processo são escritos para definir possíveis padrões de comportamento entre elementos não-determinísticos (pessoas) e ferramentas automatizadas (TULLY, 1989).

A modelagem de processo de software tem como objetivos principais (ARMENISE, 1992) (CURTIS, 1992):

- Facilitar a comunicação e compreensão entre as pessoas: a mesma representação do modelo pode ser compartilhada por todo o grupo de desenvolvimento;
- Facilitar o aperfeiçoamento do processo: através de um modelo, é possível analisar o processo e descobrir pontos onde ele pode ser melhorado;
- Reutilização: os processos não necessitam ser modelados todas as vezes que forem ser executados e, ao mesmo tempo, muitas das práticas gerenciais adotadas pelas organizações podem ser reutilizadas em diferentes contextos. Suas descrições podem ser armazenadas e reutilizadas quando necessário;
- Fornecer gerência do processo: tendo um processo definido, é possível realizar estimativas e planejamentos. Desta forma, o efeito de algumas decisões pode ser simulado e o processo definido pode ser comparado ao processo em andamento, permitindo a efetiva tomada de decisões;
- Prover orientação automatizada do processo: um processo definido permite que ferramentas automatizem algumas partes do modelo e orientem os usuários no andamento do processo;
- Prover execução automatizada: um ambiente automatizado pode controlar o comportamento do processo definido, coletar métricas e reforçar as regras para garantir a integridade do processo.

Modelos de processos de software podem ser representados através de formalismos tradicionais da Engenharia de Software como diagramas SADT, diagramas de módulos, diagramas de transição de estados, diagramas de fluxo de dados, regras de produção e modelos orientados a objeto (REIS, 1999).

2.1.2. INSTANCIAÇÃO

A instanciação é uma fase do ciclo de vida do processo aonde se apresentam características específicas relacionadas ao contexto da organização de desenvolvimento de software envolvida e aos prazos reais do projeto. São exemplos de informações necessárias para instanciação de processos: a alocação de recursos (a definição dos atores que assumirão determinados papéis), a descrição de prazos das atividades e a ferramenta a ser utilizada para executar determinada tarefa (REIS, 2003).

Modelos Instanciados (ou executáveis) são modelos prontos para execução, podendo ser submetidos à execução por uma máquina de processo. O modelo instanciado é considerado uma instância de um modelo abstrato, com objetivos e restrições específicos, envolvendo agentes, prazos, orçamentos, recursos e um processo de desenvolvimento (SILVA, 2006).

A instanciação de processos de software é a transformação de um modelo de processo em processo executável através do planejamento do cronograma e alocação de desenvolvedores e recursos para as tarefas, dentre outros ajustes. Nessa fase é necessário o conhecimento acerca do estado da organização, o contexto da execução do processo e informações de projetos anteriores. Considerando que essas informações podem ser complexas e são inter-relacionadas, a falta de atenção e suporte a essa fase pode ter como conseqüência uma alocação inadequada de recursos (REIS, 2003).

Estas informações são bastante específicas, não só por organização, mas por projetos dentro da organização. Por isso, é importante que a partir de um mesmo processo abstrato seja possível criar vários processos instanciados.

Muitas vezes não é possível instanciar todo um processo de software no início de um projeto, e nem há a garantia de que o processo vá seguir conforme o desejado. Por isso, é necessário que haja uma forma de modificar e reorganizar o processo de forma que ele passe a condizer com a realidade (DAL MORO, 2005).

2.1.3. EXECUÇÃO

A execução de processos de software é uma fase do ciclo de vida de processos de software onde atividades modeladas são realizadas tanto pelos desenvolvedores (agentes humanos) quanto automaticamente (ferramentas autônomas). Portanto, esta fase envolve questões importantes sobre planejamento, controle, monitoramento, conformidade com o processo modelado, treinamento e segurança (FEILER, 1993).

A fase de execução de um processo de software utiliza o modelo executável, obtido da instanciação dos conceitos abstratos do modelo de processo de software construído na fase de modelagem e pode ser caracterizada como a realização automatizada da construção de um software (SILVA, 2001).

Ainda na modelagem, é necessário prever as possíveis dependências entre as atividades para que se seja possível gerenciar suas dependências. Assim, o mecanismo de execução é influenciado pelo paradigma de modelagem utilizado. Por isso, a classificação de paradigmas de modelagem de processos de software também poderia ser adotada na execução. As diferenças entre um mecanismo de execução e outro utilizando o mesmo paradigma de modelagem geralmente estão na abordagem de interação com o usuário, o tratamento de segurança e autorização, a ênfase em monitoramento do processo através de métricas e a forma de interação com as outras ferramentas (REIS, 2003).

Processos de software naturalmente tendem a evoluir, devido à necessidade de estarem continuamente sendo melhorados e corrigidos devido à instabilidade do ambiente operacional. Existe um ciclo de vida para processos de software análogo ao ciclo de vida de produtos de software. As atividades deste ciclo são chamadas de meta-atividades, já o ciclo de vida é chamado de meta-processo de software (DERNIAME, 1999).

2.1.3.1. MECANISMO DE EXECUÇÃO

Automatizar o gerenciamento de uma execução de processos de software requer a utilização de ferramentas que irão manipular a descrição do processo. Têm-se para isso um mecanismo de execução de processo, que é responsável pela coordenação das atividades que devem ser realizadas por pessoas e outras ferramentas automatizadas. O suporte a automatização de processos requer uma base computacional para controlar o comportamento dentro do ambiente automatizado (SILVA, 2001).

O Mecanismo de Execução de processos executa o modelo do processo instanciado, fazendo a gerência das informações e guiando os atores conforme o modelo do processo. Um dos seus principais objetivos é garantir que o estado da execução do processo esteja consistente com o estado real da realização das tarefas.

Em (ALLILAIRE, 1992) são apresentadas algumas questões que o modelo de execução deve tratar, são elas:

- **Automação de Processo**: Apoiar a coordenação de atividades e ativar automaticamente as atividades que podem ser executadas sem intervenção humana, através de uma integração com as ferramentas do ambiente;
- **Trabalho Cooperativo**: Apoiar a cooperação de pessoas trabalhando em um projeto de software através da orientação e assistência durante todo o ciclo de desenvolvimento;
- **Monitoração**: Prover diferentes visões do estado da execução do processo, permitindo que o gerente de projetos obtenha informações sobre o andamento real das atividades;
- **Registro da história do processo**: Coletar dados da evolução do processo para permitir que o processo melhore onde houver necessidade e seja corrigido para atender novos requisitos.

2.2. ECLIPSE PROCESS FRAMEWORK

O projeto Eclipse (ECLIPSE, 2006) tem por objetivo agregar interessados em desenvolver recursos para o desenvolvimento em ambiente de códigos abertos. O principal produto, fruto dessa comunidade foi o desenvolvimento do ambiente de desenvolvimento focado na construção de ferramentas e aplicações de software. O Eclipse é atualmente uma das principais IDE Java, possuindo como características marcantes o uso da *Standard Widget Toolkit* (SWT) como biblioteca gráfica, a orientação ao desenvolvimento baseado em *plugins*, visando atender, se adaptar e dar suporte às diferentes necessidades dos seus usuários.

A plataforma Eclipse utiliza uma arquitetura aberta que possibilita que novos *plugins*, se projetados corretamente, sejam adicionados sem que haja impacto nas outras ferramentas. Essas novas ferramentas são acopladas utilizando o que é chamado de 'pontos de extensão'. A própria plataforma é construída por camadas de *plugins*, que definem um conjunto de pontos de extensão para futuras customizações. Com isso, os desenvolvedores podem focar nas tarefas específicas de seus *plugins*, sem se preocupar com a integração destes com a ferramenta (ALLILAIRE e IDRISSI, 2004).

Visando obter um conjunto completo de ferramentas capaz de ajudar o processo de desenvolvimento, um grande número de projetos é desenvolvido em paralelo dentro da comunidade. Dentre este conjunto de projetos, está o *Eclipse Process Framework* que visa

produzir um *framework* de engenharia de processo de software adaptável, suportando uma grande variedade de tipos de projeto e estilos de desenvolvimento.

Normalmente a palavra *framework* é utilizada no contexto de um ambiente que pode definir soluções completas ou parciais para um domínio particular, utilizando estruturas reutilizáveis como pontos de partida assim como blocos de construção para criar esta nova solução. *Process Framework* é algo similar para o domínio da metodologia: Define conteúdos (*method content*), processos, blocos de construção e ferramentas reutilizáveis que podem ser utilizadas para se definir processos e métodos específicos de um projeto ou organização (HAUMER, 2007).

O *Eclipse Process Framework* é uma ferramenta que pretende ajudar indivíduos e organizações a controlar e distribuir informações sobre melhores práticas no desenvolvimento de software e com isso diminuir o tempo normalmente gasto para encontrar esta informação. Apesar de existir uma grande quantidade de conhecimento disponível relacionada a desenvolvimento e processo de software, este conhecimento encontra-se disperso e não integrado em um ponto de referência organizado. A ferramenta é uma tentativa de fornecer um mecanismo e um *framework* que permita aos indivíduos e às organizações capturarem este conhecimento em um formulário que possibilite a análise de diferentes partes deste conhecimento e com isso os combinem para criar uma base de conhecimento (DEVX, 2007).

Além da ferramenta, o projeto *Eclipse Process Framework* também disponibiliza a descrição do *OpenUP/Basic*, que é um processo de software baseado no *Rational Unified Process* (RUP), cujo núcleo foi doado à comunidade Eclipse pela IBM. O *OpenUP/Basic* é um processo de software iterativo mínimo, completo e extensível. Mínimo por conter apenas conteúdo fundamental. Completo por cobrir todo o processo de desenvolvimento de um sistema. E é extensível por poder ser usado como base para qualquer processo, sendo possível adicionar ou retirar qualquer conteúdo que desejar (EPF, 2006).

Para descrever o processo, a ferramenta utiliza conceitos de disciplina de Engenharia de Software, ciclo de desenvolvimento, iteração, papéis, tarefas e descrição de cada um dos elementos. O cruzamento destes dados é feito de forma lógica, gerando uma estrutura em árvore que pode ser visualizada pelo *Browser* da ferramenta.

Por se tratar de projeto recente (a primeira versão da plataforma EPF foi disponibilizada em 30 de setembro de 2006, incluindo a ferramenta EPF Composer versão 1.0 e o OpenUP/Basic versão 0.9) e ter como objetivo a documentação, não oferece suporte a

instanciação e execução. Por ser um projeto de código aberto, existe a possibilidade de trazer novas contribuições e estas poderiam ser eventualmente evoluídas pela comunidade.

2.2.1. UNIFIED METHOD ARCHITECTURE (UMA)

O EPF possui um conjunto de componentes que juntos formam a ferramenta. Entre eles alguns merecem destaque como, por exemplo, o *Unified Method Architecture* (UMA). Este componente disponibiliza suporte ao acesso e edição dos elementos de métodos e processo armazenados em uma biblioteca. Ele define um meta-modelo descrevendo como EPF estrutura os processos e conteúdos modelados. As partes principais do UMA passaram a fazer parte do *Software Process Engineering Metamodel* (SPEM) em sua revisão mais recente, o SPEM 2.0.

O SPEM é o meta-modelo proposto pela da *Object Management Group* (OMG) para a descrição de um processo concreto de desenvolvimento de software ou uma família relacionada de processos de desenvolvimento de software. Diferente da maioria das linguagens de modelagem de processos, o SPEM aparece como uma proposta de unificação entre as diferentes metodologias propostas para modelagem de processos (GENVIGIR, 2003).

O meta-modelo UMA foi desenvolvido como uma unificação de métodos e linguagens de engenharia de processos, assim como a extensão do SPEM para a UML para a engenharia de processos de software. Como tal, ele fornece conceitos e possibilidades para todos estes modelos, os unificando de uma forma consistente e permitindo que cada um se expresse conforme suas características.

O UMA fornece uma separação clara das definições do "*Method Content*" de sua aplicação nos processos. Isto é realizado com a separação do conteúdo reutilizável (Papéis, Tarefas e Artefatos), modelado no "*Method Content*", dos processos, onde este conteúdo é referenciado.

Outro componente que merece destaque é o "*XML Export/Import*". Este componente disponibiliza o serviço para importar e exportar o conteúdo da biblioteca modelada utilizando arquivos no formato XML. Este arquivo XML é baseado no UMA e possui sua estrutura e conteúdo organizado segundo o EPF XML Schema.

3. TRABALHOS RELACIONADOS

Neste capítulo serão apresentados alguns trabalhos que suportam a definição e automação de processos de software. Estes trabalhos foram analisados com o objetivo de aprimorar o conhecimento nesta área e observar como algumas questões importantes são tratadas por estes ambientes.

3.1. AMBIENTE ExPSEE

O ambiente ExPSEE (*Experimental Process-centred Software Engineering Environment*) vem sendo desenvolvido desde 1994 pelo grupo de engenharia de software da Universidade Estadual de Maringá (UEM) e foi concebido como um ambiente de engenharia de software orientado a processos. Este ambiente tem como base de implementação a plataforma *Sun Solaris* e utiliza alguns mecanismos proprietários para realizar as suas funcionalidades. O contexto deste ambiente é especificamente o de um processo de software, apoiado por ferramentas CASE e mecanismos de modelagem e automação. É um ambiente que oferece mecanismos de apoio a modelagem e a automação de processos, tendo em vista um ciclo de vida que envolve ferramentas de apoio às suas diversas fases. Este ambiente baseia-se na definição explícita do processo por meio do qual os artefatos serão concebidos, projetados, desenvolvidos e distribuídos (OLIVEIRA JUNIOR, 2003).

O ambiente possui um gerenciador de meta-processos que oferece um controle que permite a definição de arquiteturas de processo de software e, com base nestas arquiteturas, instanciar e executar processos de software. Este gerenciador possui duas interfaces com o usuário. O *Process Architecture Builder*, responsável pela definição de arquiteturas de processo, e o *Project Manager*, responsável pela criação e gerenciamento de projetos de software, através da instanciação e execução desses processos (FANTINATO, 1999).

Através do *Process Architecture Builder*, arquiteturas de processo, compostas por tipos de objetos e seus relacionamentos podem ser definidas. Para que estas arquiteturas possam ser instanciadas em processos de software, elas devem estar completas. Uma arquitetura de processos estará completa quando todos os tipos de objetos criados também estiverem completos. Neste caso, os objetos devem possuir pelo menos um relacionamento referente a cada um dos relacionamentos definidos para o seu tipo no meta-modelo. A FIG 3.1 mostra um

exemplo de arquitetura de processo completa definida pelo *Process Architecture Builder* (FANTINATO, 1999).

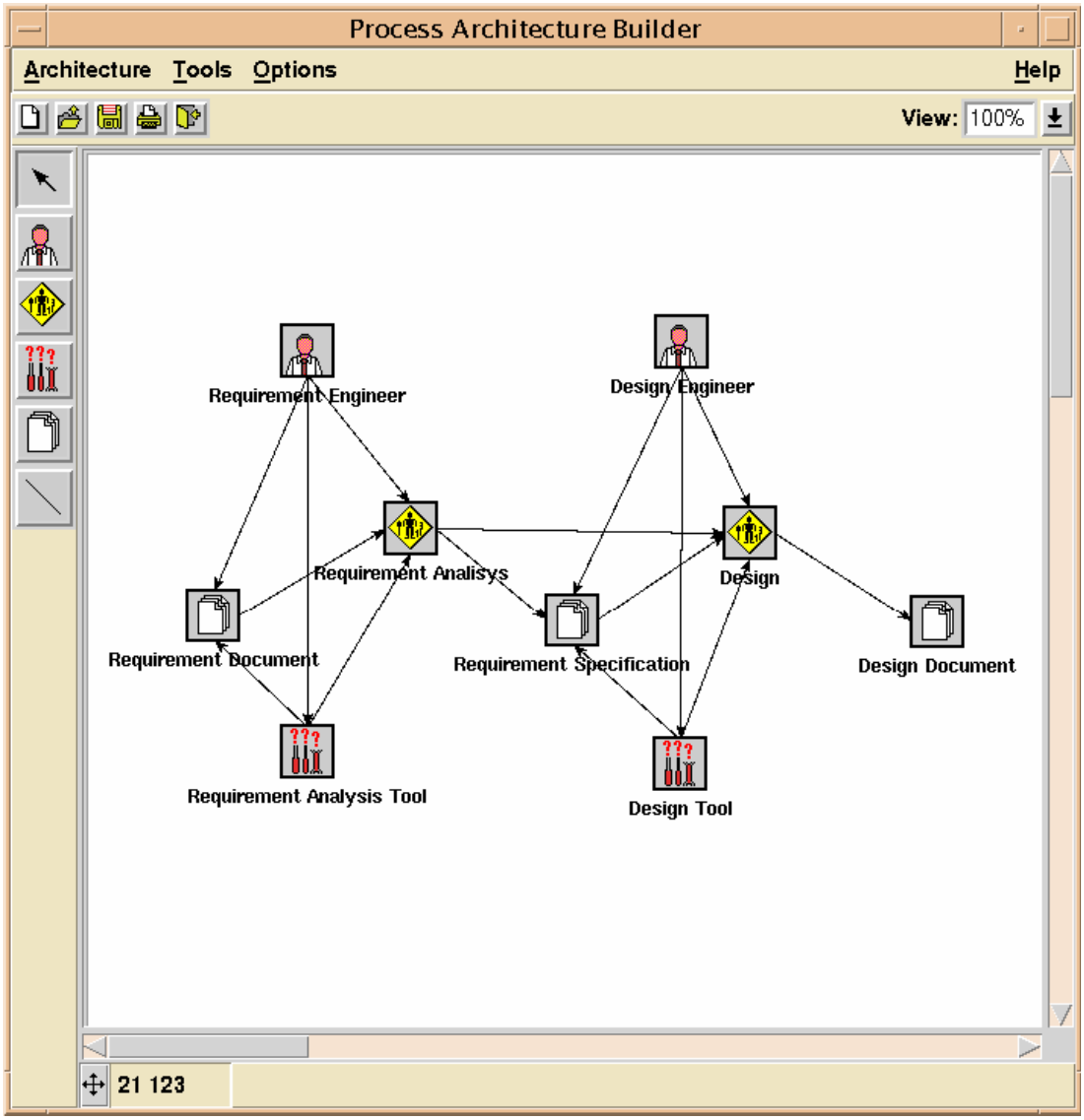


FIG 3.1 - Exemplo de uma arquitetura de processo, definida no ambiente ExPSEE.

Para cada tarefa, deve ser alocado um ator responsável pela sua execução. Os atores devem ocupar um ou mais cargos, enquanto que os cargos devem ser ocupados por um ou mais atores, definindo assim quais são os direitos e deveres que cada ator possui. Cada ator possui associado a si uma agenda onde estão relacionadas as tarefas destinadas a ele e os dados mais importantes relacionados a estas tarefas, tais como: nome, status, relacionamentos, o procedimento a ser executado, datas de início e término e carga horária destinada para sua execução (CARNIELLO, 1999).

Os atores têm a responsabilidade de alterar os estados das tarefas no ambiente, de acordo com o andamento real das mesmas. O gerente tem assim a visão do estado atual do projeto, podendo monitorar o seu andamento.

3.2. AMBIENTE WebAPSEE

O ambiente é resultado de projetos de pesquisa em andamento e foi originado a partir de uma experiência no desenvolvimento de um PSEE chamado APSEE que adota soluções inovadoras para problemas críticos relacionados com a gerência e execução automatizada de processos de software. Com o surgimento de tecnologias baseadas em *Web Services* houve uma evolução significativa no desenvolvimento de software baseado na Internet, ao facilitar a intercomunicação de sistemas através de um *middleware* aberto, multiplataforma e baseado em padrões da *Web*. A integração das idéias e modelos gerados com o projeto APSEE aliado à tecnologia dos *Web Services* resultou no ambiente WebAPSEE (LIMA, 2006).

Neste ambiente, a linguagem visual (*Process Modeling Language*) usada para modelagem de processo é a WebAPSEE-PML. Nesta linguagem foi adotada a especificação formal com abordagem de gramáticas de grafos como em (REIS, 2003). O ambiente segue o paradigma de processo orientado a atividades, descrevendo um processo como uma coleção parcialmente ordenada de atividades. Sua notação gráfica é apresentada no exemplo de modelo de processo da FIG 3.2.

Na modelagem de processos de software no WebAPSEE, os componentes de primeira ordem são as atividades (ações realizadas por desenvolvedores ou agentes de software), conexões (determinam as relações temporais e de sincronização entre atividades) e os artefatos (denominação genérica para referências de itens de software contidas em sistemas de controle de versão que são usados nos processos) (LIMA, 2006).

A WebAPSEE *Task Agenda* (no canto inferior direito da FIG 3.2) representa o cliente de interação com o desenvolvedor. Através desta o desenvolvedor visualiza os processos de software em execução nos quais está inserido como uma lista de tarefas a serem realizadas. O desenvolvedor por sua vez interage com a agenda fornecendo *feedback* sobre o andamento dessas tarefas. Esta interação é importante, pois através desta o gerente pode ter uma visão consistente do andamento de um processo (FRANÇA, 2006).

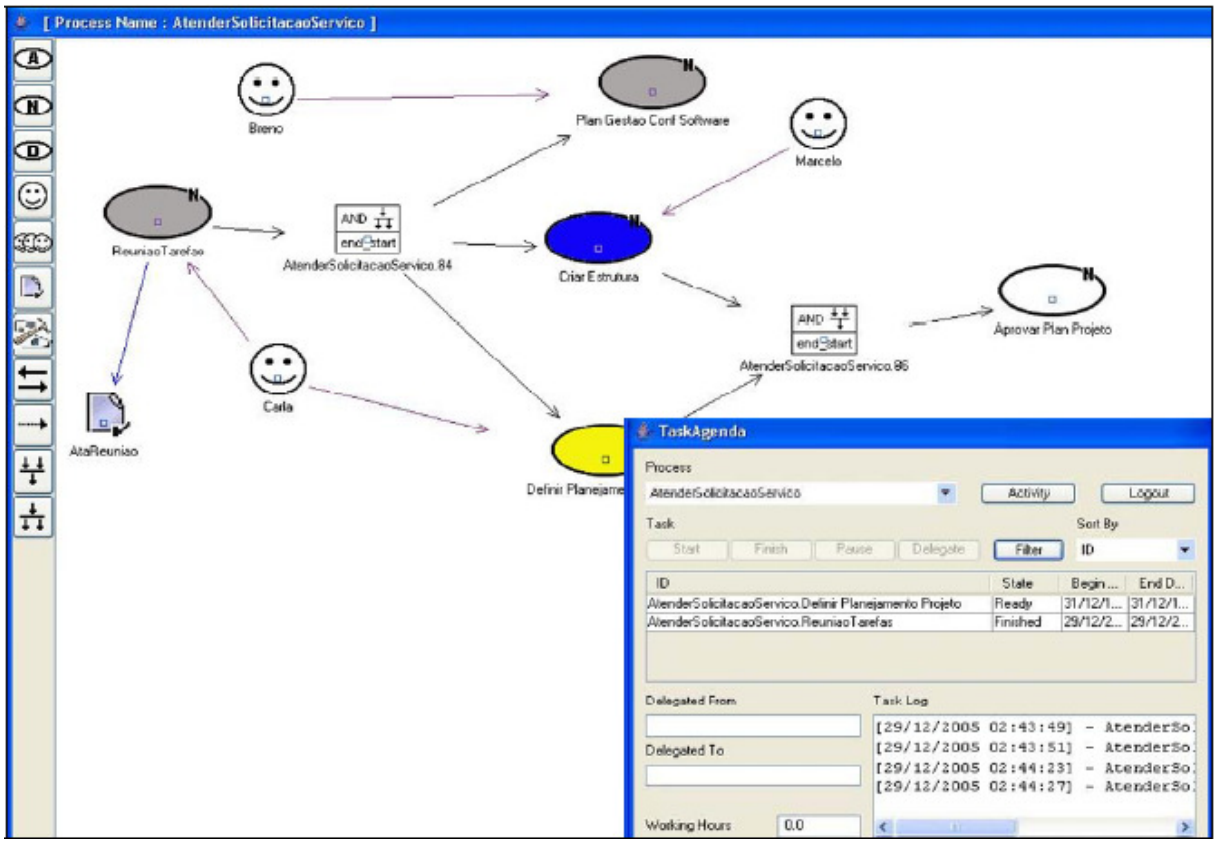


FIG 3.2 - Módulo para edição e acompanhamento do processo e Agenda de Tarefas.

3.3. ESTAÇÃO TABA

A Estação TABA é um ambiente de desenvolvimento de software que apóia a execução das atividades a serem desempenhadas em um processo de software, através de um conjunto de ferramentas integradas e repositórios contendo informações adquiridas durante a execução do processo do projeto (BERGER, 2003).

É um ambiente de desenvolvimento de software desenvolvido pela COPPE/UFRJ que inicialmente foi definido como um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de software (ADS) adequados às particularidades de processos de desenvolvimento e de projetos específicos (SILVA FILHO, 2006).

Atualmente, a Estação TABA reúne 32 ferramentas, que oferecem suporte às atividades dos processos de desenvolvimento e manutenção de software, que vão desde o apoio à instanciação de processos, atividades de planejamento, mensuração e gerência de configuração até avaliação post-mortem e apoio à captura do conhecimento, contemplando grande parte das atividades presentes no ciclo de vida do software (SILVA FILHO 2006).

A Estação TABA é um ADS centrado em processos, que agrega conhecimentos e funcionalidades desenvolvidos durante vários trabalhos de mestrado e doutorado e experiências na sua implantação em empresas. Entre eles destacamos o suporte explícito a áreas de processo do CMMI 2 e 3 e do MR MPS.BR, níveis G a C, que ajuda para uma melhor qualidade do processo e do produto (ESTOLANO, 2005).

Não foi possível avaliar exatamente como o ambiente trata o detalhamento da evolução das tarefas por parte dos atores do processo e como estes atores registram estes detalhes, que é o nosso foco principal nesta pesquisa, porque não houve a possibilidade de testar e manipular a ferramenta.

3.4. CONSIDERAÇÕES SOBRE OS TRABALHOS

Conforme apresentado anteriormente neste trabalho, um desenvolvimento de software centrado em processo possui três fases distintas em seu ciclo de vida. Inicialmente é necessário conhecer o processo que será seguido no desenvolvimento e isto é conseguido definindo o modelo abstrato do processo. Como cada projeto possui suas características específicas, para aplicar este processo em um projeto é preciso criar uma instância do processo definindo suas especificidades. Com o projeto em andamento, faz-se necessário ter um acompanhamento, garantindo que o processo esteja sendo seguido da maneira correta.

Como estas três fases (definição do modelo abstrato do processo, instanciação do modelo e acompanhamento do processo) são distintas, porém interligadas, é importante que os mecanismos responsáveis por cada uma dessas fases trabalhem de forma integrada, não exigindo grandes esforços em suas transições.

Na fase de acompanhamento, um ponto interessante abordado pelo WebAPSEE é o uso da *World Wide Web* como meio de disponibilização do ambiente, facilitando a coordenação das atividades de equipes dispersas geograficamente, como acompanhamento de prazos e consumo de recursos, além de facilitar a reutilização de boas práticas gerenciais por diferentes projetos adotados.

Uma vez definido um processo e iniciado o correspondente projeto, é esperado que algumas alterações no mesmo sejam necessárias. Isso ocorre, porque, muitas vezes, não é possível definir um processo completamente no início do projeto ou porque, durante a sua realização, novas situações surgem, sendo necessário acomodá-las. Tais alterações são ditas alterações dinâmicas do processo, ou alterações em tempo de execução (*on-the-fly*). Elas têm

o objetivo de manter o processo consistente mesmo em situações adversas como, por exemplo, a ocorrência de um risco (DAL MORO, 2005). Por conta disso, é importante que o conjunto de informações fornecidas ao gerente pelo mecanismo de acompanhamento seja detalhado o suficiente para que o mesmo conheça o estado do real processo e os motivos que o levaram a tal estado.

Para atender a estes requisitos, o mecanismo de acompanhamento deve possibilitar ao ator, não somente alterar o estado da tarefa, mas também registrar ocorrências de tudo que se passou durante a execução da atividade como: o que foi feito até o momento, dificuldades encontradas, horas alocadas, estimativa da porcentagem de conclusão, entre outros. É interessante também que o mecanismo de acompanhamento sirva como repositório dos artefatos gerados (documentos), centralizando seu armazenamento e facilitando o acesso.

Os ambientes estudados e apresentados neste capítulo não contemplam estas características em seu mecanismo de acompanhamento. Percebe-se então a ausência de um ambiente integrado para definição, instanciação e acompanhamento de processos de desenvolvimento de software que em seu mecanismo de acompanhamento ofereça este conjunto de informações relevantes ao gerente de projetos.

4. AMBIENTE INTEGRADO PARA PROCESSOS DE SOFTWARE – AIPS

Este capítulo apresentará o Ambiente Integrado para Processos de Software como uma proposta para um ambiente integrado de modelagem, instanciação e execução de processos de software. A modelagem de processos de desenvolvimento deve ser capaz de descrever alguns aspectos necessários para a coordenação das atividades. O modelo muitas vezes é utilizado apenas como documentação, ou seja, como uma forma de explicitar o processo utilizado com o objetivo de conscientizar a equipe. Existem outros casos onde ele é utilizado como guia para que seja possível acompanhar cada passo, identificar o estado do projeto e reorganizar o processo durante este acompanhamento. Este último é o caso que será considerado neste trabalho.

Na ferramenta de modelagem será permitida a definição de processos, além de permitir a modelagem de conteúdos reutilizáveis. Isto possibilita, por exemplo, a definição de diferentes processos utilizando o mesmo conjunto de tarefas, artefatos e papéis.

Para a instanciação a proposta é um mecanismo que funcione integrado a modelagem, utilizando o mesmo produto gerado pela ferramenta de modelagem como produto de entrada para o mecanismo de instanciação. Esta integração facilita a transição entre as fases e dispensa qualquer intervenção do usuário no artefato gerado. A instanciação deve permitir ainda a definição de atores para os papéis, prazos para as tarefas, iterações de atividades, entre outros. Estas informações são específicas do contexto onde o processo é aplicado e são definidas estendendo o modelo gerado pela ferramenta de modelagem.

A proposta de execução também deve trabalhar de forma integrada ao mecanismo de instanciação e permitirá o acompanhamento de um processo de software durante a sua execução, em níveis distintos para atores e gerente. Aos atores, disponibilizará as informações necessárias para que os mesmos executem suas tarefas conforme esperado e sigam o processo em questão. Ao gerente, disponibilizará uma visão geral do andamento das atividades para que o mesmo possa ter um maior controle do processo e, se necessário, realocar recursos e redefinir prazos.

O mecanismo de execução funcionará como um centralizador de informações do projeto, permitindo que nele sejam registradas as ocorrências, horas trabalhadas e artefatos gerados. Isso facilitará o acesso às informações e fará com que o grupo trabalhe em uma maior harmonia.

4.1. UM PROCESSO EXEMPLO

Com a finalidade de facilitar o entendimento, melhor ilustrar a proposta e as transformações ocorridas nas fases de modelagem, instanciação e execução, será utilizado um pequeno trecho de um processo como exemplo ao longo de todo capítulo. Este trecho se refere ao processo de definição da interface com o usuário para um determinado sistema, a partir dos requisitos definidos e da especificação do processo de trabalho do usuário.

O início deste processo exemplo se dá com a tarefa de "Definição de Requisitos", que é executada pelo "Analista Funcional". Este analista deverá recolher informações sobre os requisitos com o cliente e gerar um documento chamado "Requisitos do Usuário" com estas informações.

Ao fim da execução da tarefa e da elaboração do documento, são disparadas duas novas tarefas: "Projetar o Modelo do Processo" e "Esboçar Interface do Usuário". Esta primeira tarefa também será executada pelo "Analista Funcional" e deverá gerar o processo de trabalho do usuário. A segunda tarefa será realizada pelo "Projetista de Interface" que, utilizando o documento de "Requisitos do Usuário", deverá gerar um esboço da interface com o usuário.

De posse deste esboço, o "Projetista Técnico" poderá executar a tarefa de "Definição dos Requisitos Técnicos". Com o fim desta tarefa e com o "Processo de Trabalho" definido pelo "Analista Funcional", o "Projetista de Interface" pode gerar uma versão mais refinada da interface com o usuário. Por último, com o documento de "Refinamento da Interface do Usuário", o "Projetista Técnico" inicia a tarefa de "Construção da Aplicação".

Diferentes diagramas básicos da *Unified Modeling Language* (UML) podem ser utilizados para apresentar diferentes perspectivas de um modelo de processo de software como, por exemplo: Diagrama de Classes, Diagrama de Pacotes, Diagrama de Atividades, Diagrama de Casos de Uso, Diagrama de Seqüência e Diagrama de Atividades (SPEM, 2005). Para utilizar uma perspectiva que forneça a idéia de fluxo e ordem em que as tarefas devem ocorrer, será utilizado o Diagrama de Atividades. Com este diagrama, é possível indicar a seqüência das tarefas, as suas dependências, além do início e do fim do processo.

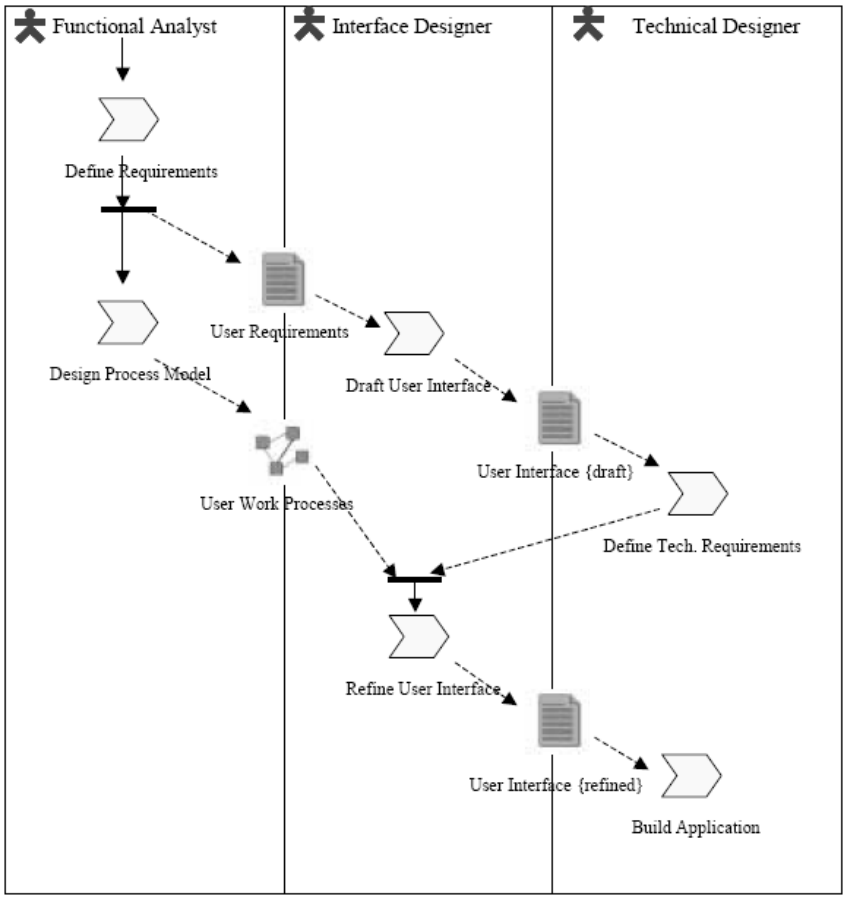


FIG 4.1 - Exemplo de um processo modelado utilizando o meta-modelo SPEM.

4.2. MODELAGEM

A proposta apresentada neste trabalho utiliza a ferramenta *Eclipse Process Framework* para realizar a modelagem de processos de software. Este trabalho optou por utilizar o EPF para dar suporte à fase de modelagem porque a ferramenta dispõe de um extenso conjunto de recursos para definição e manipulação de processos de software, como por exemplo, o recurso de reutilização dos elementos modelados. Outro fator que contribuiu para a escolha foi o fato de utilizar para modelagem visual dos processos os elementos da especificação do SPEM. Entre outros fatores, destacamos que a ferramenta ainda não oferece suporte a instanciação e execução de processos e é um projeto de código aberto, o que possibilita que novas contribuições sejam feitas e com isso a ferramenta seja continuamente aprimorada.

A ferramenta apoiará a definição de todos os elementos que compõem o processo e a definição do processo propriamente dito, atendendo aos requisitos propostos. Possui ainda um recurso para geração de um produto em XML representando o modelo, que possibilitará ler e manipular os dados modelados futuramente.

Para se modelar um processo de software é preciso, inicialmente, identificar todos os elementos que compõem este processo como: os artefatos, os papéis e as tarefas. Após identificar os elementos, é necessário coletar algumas informações específicas de cada um deles, para que estas informações sejam explicitadas no modelo e futuramente possam ser analisadas por quem desejar. No EPF cada elemento possui um formulário de cadastro bastante extenso e completo, o que permite que estes elementos sejam bem detalhados e possam orientar suficientemente os atores.

Um elemento pode aparecer mais de uma vez no processo, então eles devem ser modelados em uma área específica para que possam ser referenciados quantas vezes forem necessárias. Desta forma, supondo que uma tarefa seja executada durante a fase de análise e posteriormente na fase de projeto, esta tarefa seria modelada uma única vez, porém referenciada duas vezes no mesmo processo. Estes elementos são modelados no "*Method Content*", que funciona como um repositório para os elementos e os torna visíveis para a modelagem do processo.

Assim como os elementos, pequenas partes do processo também podem ser modeladas separadamente e referenciadas posteriormente. Estas pequenas partes do processo podem representar, por exemplo, fases do processo de desenvolvimento agrupando uma seqüência de tarefas ou mesmo um conjunto de tarefas que sofre iteração.

No EPF, estes pequenos processos devem ser modelados no repositório "*Capability Patterns*" que se encontra na área reservada para processos. Cada um deles é modelado como um modelo completo e pode possuir seu diagrama de atividades, referenciando as tarefas modeladas no "*Method Content*". Este pequeno conjunto de tarefas, chamado de atividade, permite a modularização do processo e facilita o entendimento e acompanhamento das tarefas no modelo.

Desta forma, no EPF, é possível criar um modelo do processo de software fazendo referências a tarefas do "*Method Content*" ou a atividades do "*Capability Patterns*". Este modelo deve ser criado no repositório "*Delivery Process*", que também se encontra na área reservada para processos. Para comportar modelos para ciclos de vida de desenvolvimento de software iterativo, deve-se indicar que determinada atividade é ou não uma iteração, para que na instanciação seja possível definir o número de iterações e na execução estas repetições sejam levadas em consideração.

Tendo finalizado a modelagem do processo, é necessário gerar o artefato que representa o modelo para que este seja instanciado. Dentre os vários componentes que fazem parte do EPF,

existe um que possibilita a exportação do modelo como um arquivo XML, que segue a especificação do *Unified Method Architecture*. Este artefato conterá não só o processo, mas também todo o conteúdo reutilizável.

4.2.1. UM PROCESSO EXEMPLO: MODELAGEM

Antes de se modelar um processo no EPF, é preciso compreender a forma como a ferramenta está estruturada. O princípio fundamental do *Eclipse Process Framework* é ter os elementos reutilizáveis modelados separadamente do processo.

Ao se criar um novo modelo, ele é iniciado vazio e exibe a sua estrutura padrão. Esta estrutura possui o "*Method Content*", onde modelamos os elementos reutilizáveis, e o "*Processes*", onde modelamos os processos. Uma das subdivisões do "*Method Content*" é o "*Content Packages*" que por sua vez possui subdivisões como: "*Roles*", "*Tasks*" e "*Work Products*". Já o "*Processes*" é subdividido em "*Capability Patterns*" e "*Delivery Processes*". Esta estrutura completa pode ser observada na FIG 4.2.

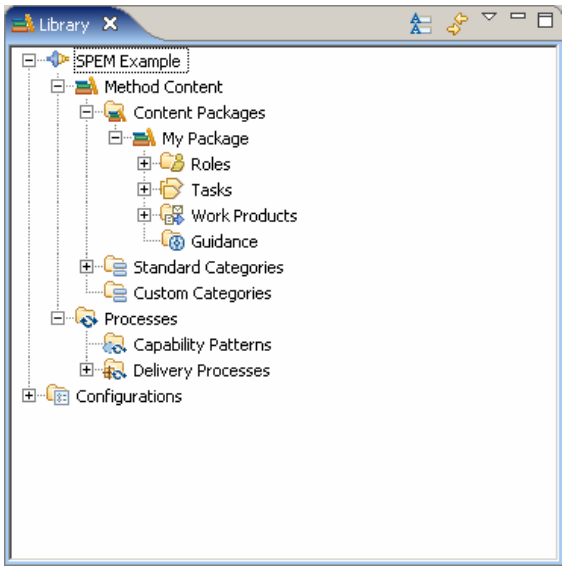


FIG 4.2 – Estrutura do EPF.

Inicialmente deve-se identificar cada um dos elementos presentes no processo para que os mesmos possam ser modelados na ferramenta. Então, analisando a FIG 4.1 são identificados os seguintes elementos:

- Atores: "*Functional Analyst*", "*Interface Designer*" e "*Technical Designer*";

- Tarefas: "*Define Requirements*", "*Design Process Model*", "*Draft User Interface*", "*Refine User Interface*", "*Define Tech. Requirements*" e "*Build Application*";
- Artefatos: "*User Requirements*", "*User Work Processes*", "*User Interface {draft}*" e "*User Interface {refined}*".

Ainda no diagrama, é possível identificar que atores realizam cada tarefa, devido a sua estrutura de raias (*Swimlanes*), além dos artefatos de saída e de entrada de cada uma das tarefas.

Ciente dos elementos presentes no processo e de suas principais características pode-se começar a defini-los na ferramenta seguindo o modelo conceitual do "*Method Content*". Este modelo conceitual consiste em ter-se um conjunto de papéis que são responsáveis por artefatos. Estes artefatos são produzidos por tarefas que são realizadas por papéis e possuem artefatos de entrada e de saída. Uma definição deste modelo conceitual é exemplificada na FIG 4.3.

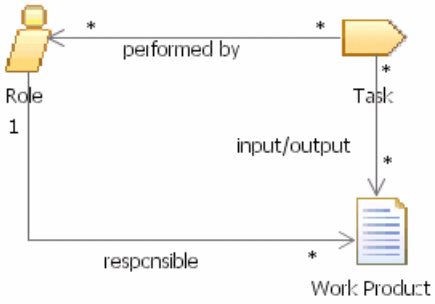


FIG 4.3 – Modelo conceitual simplificado do "*Method Content*".

Após a modelagem dos elementos reutilizáveis pode-se modelar o processo e para isso é criado um novo "*Delivery Process*". Existe a opção de se modelar a partir de um diagrama de atividades ou a partir do *Work Breakdown Structure* (WBS). Qualquer mudança feita em um dos modelos é refletida no outro. Em ambos os casos, deve-se referenciar as tarefas que foram modeladas anteriormente em algum pacote do "*Method Content*".

Pode-se observar o processo exemplo modelado como um WBS na FIG 4.4, como um diagrama de atividades na FIG 4.5 e a visão consolidada na FIG 4.6, que possibilita a visão de papéis e artefatos por tarefa.

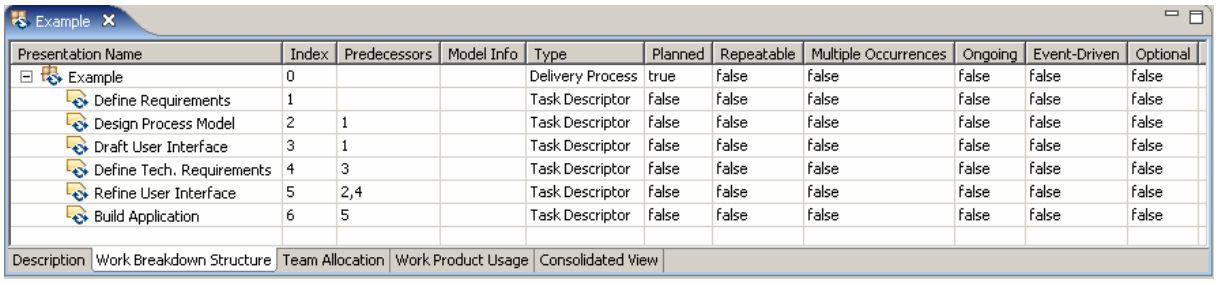


FIG 4.4 – WBS do processo exemplo modelado no EPF.

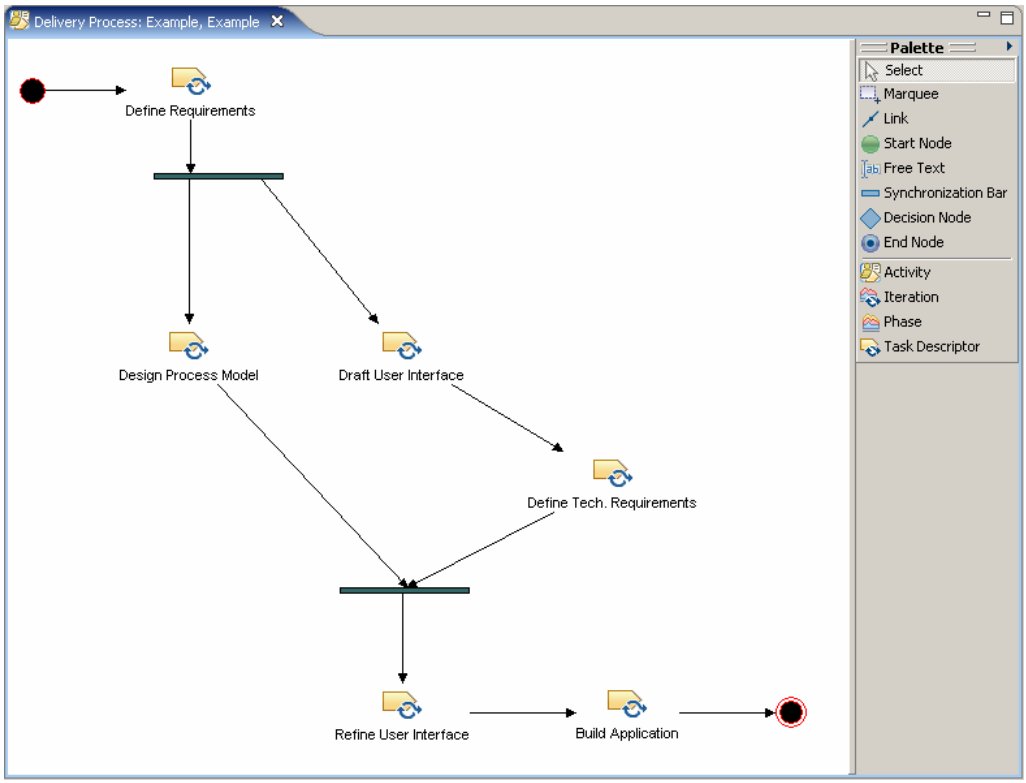
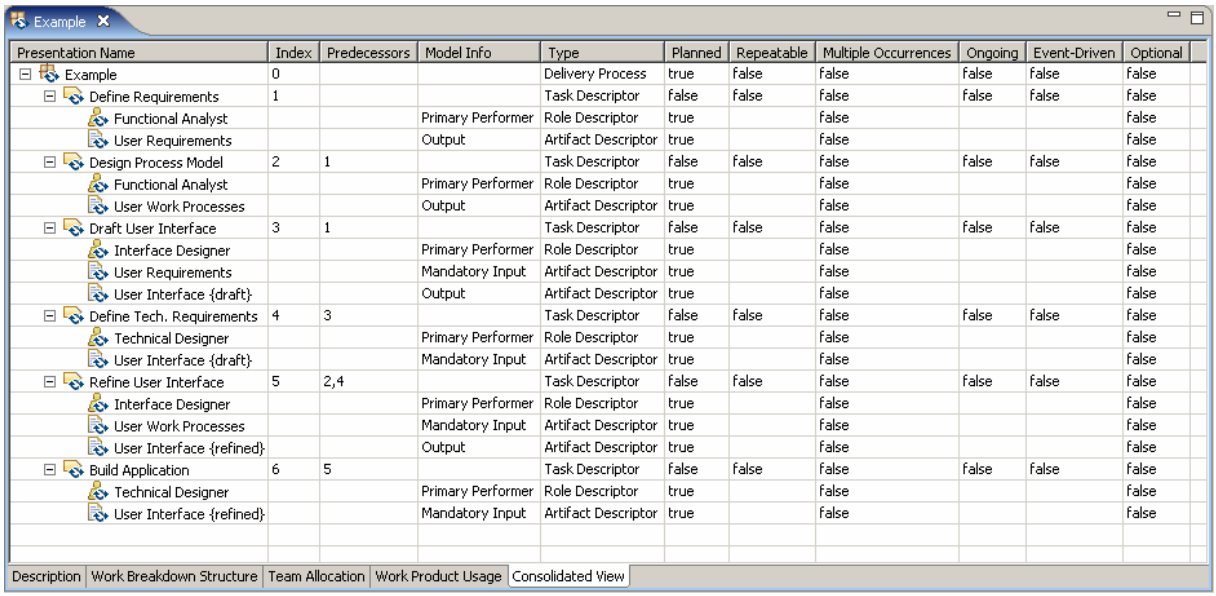


FIG 4.5 – Diagrama de Atividades do processo exemplo modelado no EPF.



Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeatable	Multiple Occurrences	Ongoing	Event-Driven	Optional
Example	0			Delivery Process	true	false	false	false	false	false
Define Requirements	1			Task Descriptor	false	false	false	false	false	false
Functional Analyst			Primary Performer	Role Descriptor	true		false			false
User Requirements			Output	Artifact Descriptor	true		false			false
Design Process Model	2	1		Task Descriptor	false	false	false	false	false	false
Functional Analyst			Primary Performer	Role Descriptor	true		false			false
User Work Processes			Output	Artifact Descriptor	true		false			false
Draft User Interface	3	1		Task Descriptor	false	false	false	false	false	false
Interface Designer			Primary Performer	Role Descriptor	true		false			false
User Requirements			Mandatory Input	Artifact Descriptor	true		false			false
User Interface {draft}			Output	Artifact Descriptor	true		false			false
Define Tech. Requirements	4	3		Task Descriptor	false	false	false	false	false	false
Technical Designer			Primary Performer	Role Descriptor	true		false			false
User Interface {draft}			Mandatory Input	Artifact Descriptor	true		false			false
Refine User Interface	5	2,4		Task Descriptor	false	false	false	false	false	false
Interface Designer			Primary Performer	Role Descriptor	true		false			false
User Work Processes			Mandatory Input	Artifact Descriptor	true		false			false
User Interface {refined}			Output	Artifact Descriptor	true		false			false
Build Application	6	5		Task Descriptor	false	false	false	false	false	false
Technical Designer			Primary Performer	Role Descriptor	true		false			false
User Interface {refined}			Mandatory Input	Artifact Descriptor	true		false			false

FIG 4.6 – Visão Consolidada do processo exemplo modelado no EPF.

Para exemplificar o produto exportado pela ferramenta, será apresentado um pequeno trecho que compõe este XML, onde será possível observar como ele está estruturado. Todos os elementos modelados como conteúdo reutilizável são apresentados no início do arquivo dentro do pacote criado para comportar estes elementos. Em seguida, os processos são descritos e pode-se observar que para cada elemento presente em um processo é criado um *Descriptor* para este elemento, que referencia o elemento modelado no conteúdo reutilizável. Abaixo, o trecho resumido do produto. Para fins de consulta, o texto completo encontra-se no anexo A.

```
<?xml version="1.0" encoding="UTF-8"?>
<uma:MethodLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" … >
  <MethodPlugin name="SPEM Example" briefDescription="" id="_srD1wAilEdyJg53g-AgF8w"… >
    <MethodPackage xsi:type="uma:ContentCategoryPackage" name="ContentCategories" … />
    <MethodPackage xsi:type="uma:ContentPackage" name="My Package" … >
      <ContentElement xsi:type="uma:Role" name="Functional Analyst" … />
      <ContentElement xsi:type="uma:Artifact" name="User Requirements" … />
      <ContentElement xsi:type="uma:Task" name="Define Requirements" … >
        <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
        <Output>_5f9roAilEdyJg53g-AgF8w</Output>
      </ContentElement>
    </MethodPackage>
    <MethodPackage xsi:type="uma:ProcessComponent" name="Example" … >
      <Process xsi:type="uma:DeliveryProcess" name="Example" briefDescription="" … >
        <BreakdownElement xsi:type="uma:TaskDescriptor" name="Define Requirements" … >
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <Task>_Ag70AAimEdyJg53g-AgF8w</Task>
          <PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
          <Output>_mgewkQiqEdyTutl_HeDilg</Output>
        </BreakdownElement>
        <BreakdownElement xsi:type="uma:RoleDescriptor" name="Functional Analyst" … >
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <Role>_xUCzEAilEdyJg53g-AgF8w</Role>
        </BreakdownElement>
        <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User Requirements"… >
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <WorkProduct>_5f9roAilEdyJg53g-AgF8w</WorkProduct>
        </BreakdownElement>
      </Process>
    </MethodPackage>
  </MethodPlugin>
  <MethodConfiguration name="SPEM Example" id="_t_stcAilEdyJg53g-AgF8w" … >
  </MethodConfiguration>
</uma:MethodLibrary>
```

4.3. INSTANCIAÇÃO

Esta é a fase do ciclo de vida do processo onde se apresentam características específicas relacionadas ao contexto da organização de desenvolvimento de software envolvida e aos prazos reais do projeto. Algumas informações são necessárias na instanciação de processos como: a alocação de recursos (a definição dos atores que assumirão determinados papéis), a descrição de prazos das atividades e a ferramenta a ser utilizada para executar determinada tarefa (REIS, 2003).

No modelo abstrato gerado na fase anterior têm-se as informações gerais do processo como os papéis, as tarefas, os artefatos e a forma como estes elementos estão relacionados. Estas informações são obtidas a partir da ferramenta de modelagem já existente, o EPF, mas não são suficientes para que a execução do processo seja automatizada e que haja um controle do andamento de um projeto onde este processo esteja sendo utilizado. Por esse motivo, um novo conjunto de informações deve ser agregado ao modelo abstrato do processo. Neste ponto entra a primeira contribuição deste trabalho que irá possibilitar a definição de um novo

conjunto de informações, relacionadas às informações existentes no modelo abstrato, mas que são específicas para o contexto onde o processo será aplicado.

Na instanciação, o modelo do processo deve ser interpretado para que sejam identificados os elementos que compõem este processo. Elementos como os papéis, as tarefas e os artefatos deverão receber um tratamento especial para que cada um deles seja instanciado, ou seja, para que cada um deles agregue novas informações que dirão respeito ao contexto onde o processo será aplicado.

Nesta proposta, a interpretação do modelo deve ser feita de forma integrada à ferramenta de modelagem, o que significa dizer que o artefato exportado pelo EPF deve ser o mesmo que será utilizado pelo mecanismo de instanciação sem que haja necessidade de qualquer adaptação no artefato por parte do usuário. Este mecanismo deve ser implementado como um *plugin* para o Eclipse, mantendo assim o usuário dentro do mesmo ambiente e cooperando para que esta transição entre a fase de modelagem e instanciação ocorra de forma tranqüila.

Um mesmo processo de desenvolvimento pode ser aplicado em diversos projetos distintos, mas não deve haver a necessidade de se gerar um modelo para cada um destes projetos, ou seja, deve haver a possibilidade de reutilização de um mesmo modelo do processo para geração de várias instâncias dele. Isto é permitido no mecanismo proposto por meio da importação do artefato gerado pela modelagem, pois ao se importar o XML do modelo seus dados são carregados e o seu conteúdo é replicado para o artefato que será gerado na instanciação.

Neste conjunto de dados que são carregados do modelo, estão os papéis que fazem parte do processo e que são encarregados de executar tarefas. Estes papéis são desempenhados por atores que devem ser devidamente cadastrados e associados aos papéis que representem suas responsabilidades no projeto. Como é comum encontrar estruturas de equipes em que um papel é desempenhado por diversos atores e também um mesmo ator desempenhando mais de um papel, a instanciação deve permitir que as atribuições de papéis a atores também seja feita desta forma e com isso refletir a realidade da equipe.

No cadastro de um ator, além de definir os seus papéis, outras duas informações são necessárias: seu nome e seu e-mail. O nome será importante para que o ator possa ser identificado na fase de execução. O e-mail também será utilizado na fase de execução, porém como meio de comunicação, pois o mecanismo de execução pode, eventualmente, gerar notificações automáticas ou mesmo o gerente pode desejar enviar alguma mensagem aos integrantes da equipe.

O número de iterações para cada atividade existente no modelo é outra informação que varia de projeto para projeto o que justifica que este dado seja definido na fase de instanciação. Desta forma, o mecanismo de instanciação deve permitir que para cada atividade presente no processo seja possível definir o número de vezes que esta atividade deve ser executada e com isso abranger a instanciação de processos, não só em cascata, mas também iterativos.

Tendo definido as iterações das atividades, e por conseqüência o conjunto de tarefas que deverão ser realizadas durante a execução do processo, torna-se possível definir quando cada uma destas tarefas deve ser iniciada e finalizada. Para que esta definição seja feita, o mecanismo de instanciação apresenta o processo como uma estrutura de árvore onde são exibidas as atividades e as tarefas que a compõem. O usuário deve selecionar a tarefa desejada e definir o seu prazo.

As datas de início e fim das tarefas são importantes por indicarem a ordem em que as tarefas devem ser executadas, e por definirem não só a estimativa de prazo para a execução de cada tarefa, mas o prazo do projeto como um todo. Estas datas também permitem que na execução seja possível saber se uma tarefa ou mesmo o projeto se encontra atrasado, para que o gerente possa tomar as devidas providências.

Os últimos elementos do modelo que faltam ser instanciados são os artefatos gerados pelas tarefas. Estes artefatos podem ser de diversos tipos, como um documento de texto, um diagrama da UML, entre outros, e por isso também necessitam de informações além das existentes no modelo abstrato. Uma destas informações é o conjunto de ferramentas que podem ser utilizadas para gerar determinado artefato. Este dado é fundamental para que os artefatos gerados possam ser examinados por todos os participantes do projeto e não haja conflito entre os tipos de arquivos gerados e as ferramentas que são disponibilizadas para os demais integrantes da equipe.

Se for considerado que estes artefatos gerados pelas tarefas serão armazenados e utilizados não apenas pela equipe de desenvolvimento, mas também pelo cliente do projeto é possível compreender que para cada projeto e clientes diferentes haja não só um conjunto de ferramentas distintas, mas também de *templates* distintos para geração destes documentos. Visto isso, para cada artefato, o mecanismo de instanciação permite que seja indicado um *template* que deverá ser usado como base para geração do mesmo.

O mecanismo de instanciação apóia o usuário na definição de informações importantes para uma futura execução e acompanhamento do processo como, por exemplo, informações

que dizem respeito a organização da equipe, ciclo de vida do processo além de prazo e controle das tarefas. Integrado à ferramenta de modelagem, o mecanismo gera um produto que possui não só as informações do modelo abstrato do processo, mas agrega a ele um novo conjunto de dados tornando o modelo executável. Um modelo de como este produto gerado está estruturado é apresentado a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<uma:MethodLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... >
...
(CONTEÚDO DO MODELO)
...
  <Instancia>
    <Actor name="NOME_ATOR" email="EMAIL_ATOR" id="ID_ATOR" >
      <Role>ID_PAPEL_1</Role>
      <Role>ID_PAPEL_2</Role>
      ...
    </Actor>
    <TaskInstance name="NOME_TAREFA" id="ID_TAREFA_INSTANCIADA"
     idDescriptor="ID_DESCRIPTOR_TAREFA" startDate="DATA_INÍCIO"
     finishDate="DATA_FIM">
        <Task>ID_TAREFA_MODELO</Task>
    </TaskInstance>
    <Artifact name="NOME_ARTEFATO" id="ID_ARTEFATO_MODELADO"
     templatePath="CAMINHO_TEMPLATE">
        <Tool>NOME_FERRAMENTA_1</Tool>
        <Tool>NOME_FERRAMENTA_2</Tool>
        ...
    </Artifact>
  </Instancia>
</uma:MethodLibrarv>
```

4.3.1. UM PROCESSO EXEMPLO: INSTANCIAÇÃO

Um dos recursos disponíveis no *Eclipse Process Framework* é a exportação do processo modelado para um arquivo XML. É justamente o XML gerado por esta exportação que servirá como artefato de entrada para o mecanismo de instanciação.

O mecanismo criado é um *plugin* que funciona como um editor multi-páginas para arquivos com a extensão “instancia”. Portanto, com o *plugin* instalado no Eclipse, os arquivos com a extensão “instancia” poderão ser editados com por este mecanismo, que irá incluir as informações específicas da instanciação no arquivo XML, tornando este arquivo num modelo instanciado do processo.

Visto isso, para se instanciar um processo, inicialmente, deve-se criar um arquivo “instancia” e importar para dentro dele o XML do processo modelado. Para realizar esta

importação basta abrir a aba “Importar” e localizar o arquivo XML em seu computador. Um exemplo da tela de importação é ilustrado na FIG 4.7.

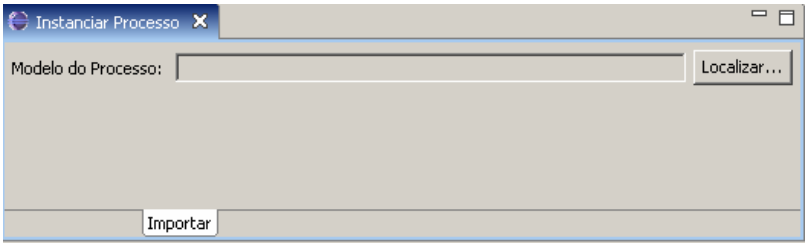


FIG 4.7 – Tela para importação do modelo abstrato.

Após importar o arquivo, o *plugin* carrega todos os dados necessários a partir do modelo para que as demais informações possam ser definidas. Pode-se então abrir a aba “Atores”, para se definir quais atores desempenharão os três papéis existentes no processo exemplo. A tela para definição dos atores é ilustrada na FIG 4.8, onde é possível observar como esta definição é feita.

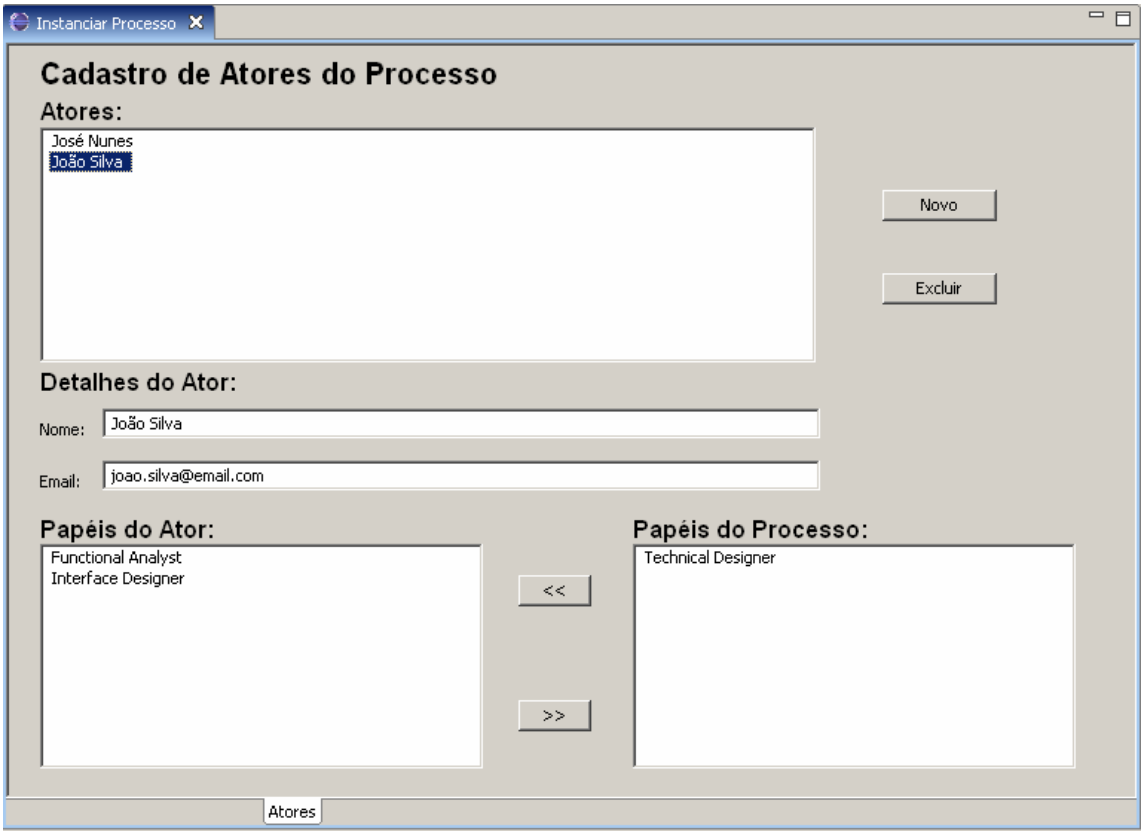


FIG 4.8 – Tela para definição de atores.

No XML criado pela ferramenta de modelagem, os papéis são estruturados da seguinte forma:

```
<ContentElement xsi:type="uma:Role" name="Functional Analyst"
  briefDescription="" id="_xUCzEAilEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="Functional Analyst"
  variabilityType="na"/>
<ContentElement xsi:type="uma:Role" name="Interface Designer"
  briefDescription="" id="_0bj9oAilEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="Interface Designer"
  variabilityType="na"/>
<ContentElement xsi:type="uma:Role" name="Technical Designer"
  briefDescription="" id="_2mmR0AilEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="Technical Designer"
  variabilityType="na"/>
```

Então, o *plugin* inclui no XML elementos do tipo "*Actor*" que possuem como filhos os papéis ("*Role*"). Abaixo um trecho do arquivo indicando que João Silva possui os papéis *Functional Analyst* e *Interface Designer*, e que José Nunes possui o papel *Technical Designer*.

```
<Actor name="João Silva" email="joao.silva@email.com"
  id="_dVHsXB3bdZ30UeX3xwou4M" >
      <Role>_xUCzEAilEdyJg53g-AgF8w</Role>
      <Role>_0bj9oAilEdyJg53g-AgF8w</Role>
</Actor>

<Actor name="José Nunes" email="jose.nunes@email.com"
  id="_9YbzSbBo5zFkSReoOhcV52" >
      <Role>_2mmR0AilEdyJg53g-AgF8w</Role>
</Actor>
```

Para definir as datas de início e fim para cada uma das tarefas, deve-se abrir a aba "Tarefas". Será exibida uma tabela em estrutura de árvore semelhante ao WBS do EPF, onde ao se selecionar uma tarefa os campos localizados abaixo da tabela são liberados para edição. As datas de início e fim são definidas selecionando o dia em um calendário que é exibido ao clicar no botão "<" ao lado do campo desejado.

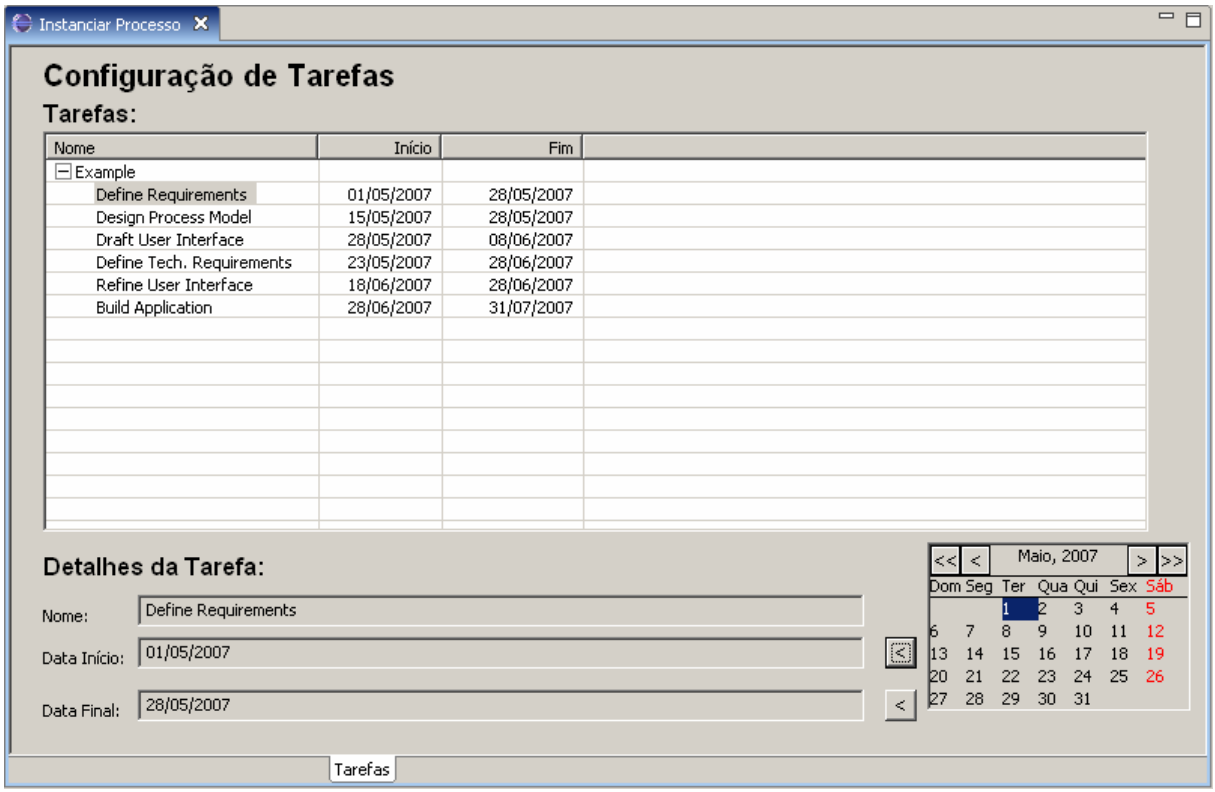


FIG 4.9 – Tela para definição de prazos e ferramentas para as tarefas.

Abaixo, um exemplo do trecho XML da tarefa "*Define Requirements*" que indica, além das informações da tarefa, os papéis que a executam e seus artefatos de entrada e saída.

```
<ContentElement xsi:type="uma:Task" name="Define Requirements"
  briefDescription="" id="_Ag70AAimEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="Define Requirements"
  variabilityType="na">
      <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
      <Output>_5f9roAilEdyJg53g-AgF8w</Output>
</ContentElement>
```

Mas como a tarefa é modelada no "*Method Content*", isto a torna reutilizável podendo ser utilizada mais de uma vez em um mesmo processo. Então é criado um elemento no XML que representa a tarefa dentro do processo. O trecho em XML abaixo exemplifica a tarefa "*Define Requirements*" existente no processo "*Example*" que foi modelado.

```
<BreakdownElement xsi:type="uma:TaskDescriptor" name="Define Requirements"
  briefDescription="" id="_xDFJcAimEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="Define Requirements"
  hasMultipleOccurrences="false" isOptional="false" isPlanned="false"
  prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false"
  isSynchronizedWithSource="true">
      <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
      <Task>_Ag70AAimEdyJg53g-AgF8w</Task>
      <PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
      <Output>_mgewkQiqEdyTutl_HeDilg</Output>
</BreakdownElement>
```

A instância da tarefa que é criada pelo *plugin* referencia também este elemento, além de indicar as datas que foram definidas. Abaixo, um exemplo do XML da instância da tarefa acima.

```
<TaskInstance name="Define Requirements" id="_jDrPUQimEdyJg53g-
  AgF8w,_xDFJcAimEdyJg53g-AgF8w" idDescriptor="_xDFJcAimEdyJg53g-AgF8w"
  startDate="01/05/2007" finishDate="28/05/2007">
      <Task>_Ag70AAimEdyJg53g-AgF8w</Task>
</TaskInstance>
```

Por último, os artefatos serão instanciados, definindo o conjunto de ferramentas que podem ser utilizadas para sua elaboração e um caminho para um *template* do artefato. Este caminho pode ser um endereço de rede ou uma URL para o arquivo. A FIG 4.10 exibe uma tela ilustrando como esta definição deve ser feita.

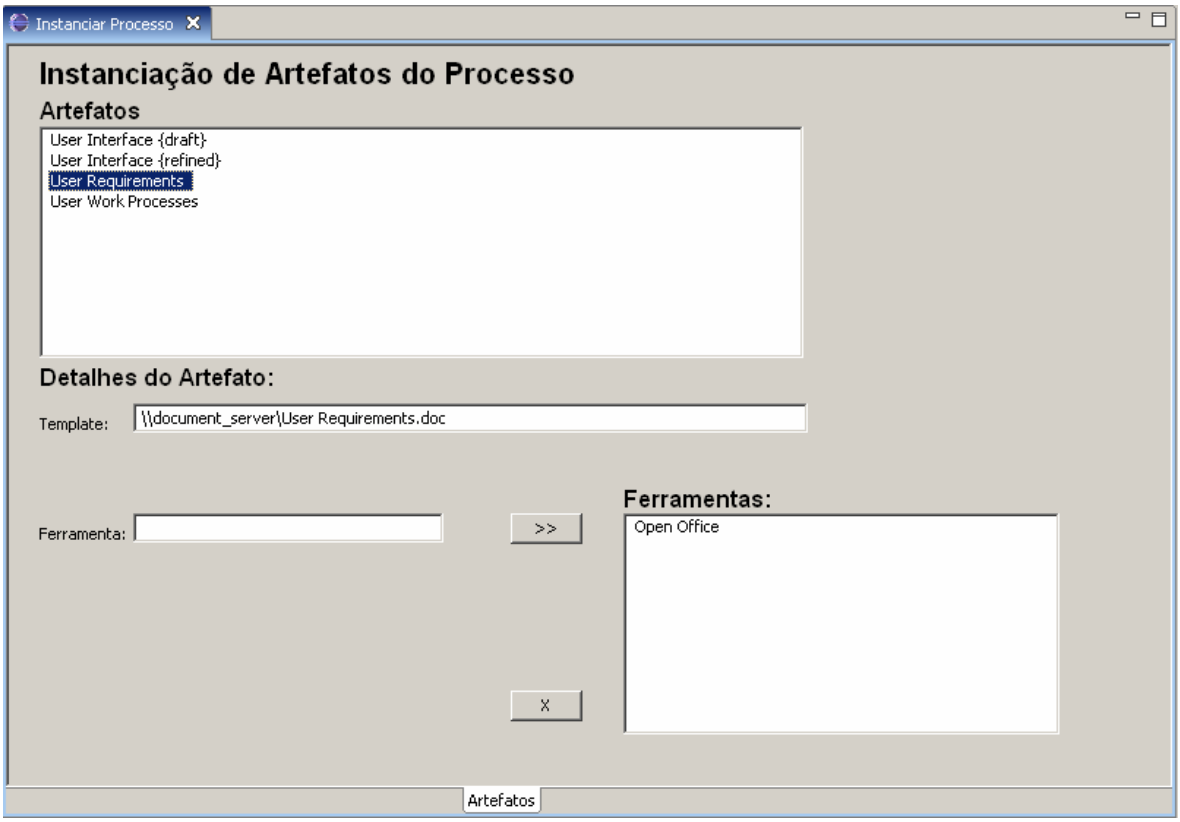


FIG 4.10 – Tela para instanciação de artefatos.

No modelo, os artefatos são representados no arquivo XML da seguinte forma:

```
<ContentElement xsi:type="uma:Artifact" name="User Requirements"
  briefDescription="" id="_5f9roAilEdyJg53g-AgF8w" orderingGuide=""
  suppressed="false" presentationName="User Requirements"
  variabilityType="na"/>
```

O *plugin* cria uma instância para o artefato, referenciando o artefato do modelo e indicando suas novas informações. Abaixo um trecho do XML gerado.

```
<Artifact name="User Requirements" id="_5f9roAilEdyJg53g-AgF8w"
  templatePath="\\document_server\User Requirements.doc">
      <Tool>Open Office</Tool>
</Artifact>
```

Todos estes elementos são incluídos no fim do arquivo como filhos de um elemento "<Instancia>" também criado pelo *plugin*. O restante do arquivo é mantido para que os dados

do modelo possam ser carregados na fase de execução. Para fins de consulta, a composição deste arquivo encontra-se no anexo B.

4.4. EXECUÇÃO

A execução de processos de software é a fase que permite o controle e o monitoramento do andamento do processo. Esta fase utiliza informações presentes no modelo executável do processo para guiar atores e gerentes durante sua execução. Com as informações provenientes da instanciação adicionadas às informações do modelo abstrato, é possível conhecer: os atores do processo, os papéis que desempenham, as tarefas que devem executar, o prazo em que as tarefas devem ser executadas, além dos artefatos utilizados e gerados por estas tarefas.

Este mecanismo de execução deve ser a principal fonte de informações para os atores e gerentes, além de ser um repositório para os artefatos. Por isso é importante que todos os participantes da equipe possam ter acesso a ele, independente de onde os integrantes se encontrem fisicamente. Então este deve ser um mecanismo Web onde, para cada ator do processo, seja criada uma conta de acesso para o perfil ao qual faça parte na equipe.

Como todas estas informações estão presentes no produto gerado pela instanciação, as fases de modelagem, instanciação e execução devem ser integradas. Assim, como as fases de modelagem e instanciação, as fases de instanciação e execução devem trabalhar de forma integrada garantindo que todo o ciclo se comporte desta maneira. Portanto, o produto gerado pelo mecanismo de instanciação deve ser o mesmo que será utilizado como fonte de dados inicial pelo mecanismo de execução.

Estes dados serão importantes para guiar o ator durante a execução do processo, pois com eles será possível apresentar a cada ator sua agenda de compromissos com datas e detalhes para as tarefas alocadas. A disponibilização destas informações é importante para garantir um bom andamento do projeto, pois conscientiza os atores de suas responsabilidades e expectativas do gerente em relação a prazos e produtos gerados. Neste conjunto de detalhes relacionados as tarefas e que são disponibilizados ao ator, estão informações provenientes da modelagem e também da instanciação.

Da modelagem, destacam-se a descrição, os passos a serem seguidos e os artefatos de entrada e saída para cada tarefa, importantes por esclarecerem ao ator a forma como esta deve ser executada, indicarem fonte de dados como artefatos de entrada e produtos a serem gerados como artefatos de saída. Da instanciação são importantes as informações de data de início e

fim das tarefas por apontarem não só quando a tarefa deve iniciar, mas porque permitem ao ator conhecer a estimativa de prazo para sua conclusão.

Ao listar os artefatos de entrada e saída para uma determinada tarefa, o mecanismo permite que o ator possa visualizar os detalhes destes artefatos. No que diz respeito aos artefatos de entrada, destacam-se como dados importantes a serem disponibilizados: a ferramenta utilizada para gerar este artefato e o próprio artefato. Tais informações são importantes, pois se a execução de uma tarefa requer que algum produto tenha sido gerado previamente. É interessante que o executor desta tarefa possa analisar este artefato. Desta forma, o ator tem seu acesso a este documento facilitado além da indicação de qual ferramenta deve ser utilizada para que o mesmo seja analisado.

Considerando os artefatos de saída, além da ferramenta a ser utilizada para gerar o artefato, o mecanismo disponibiliza o link para acesso ao *template* deste documento, informações importantes para que os documentos sigam o padrão definido. Após gerar o artefato, o ator deve utilizar o próprio mecanismo de execução como repositório de documentos, onde é possível armazenar diversas versões para cada, inclusive versões inacabadas. Ao centralizar o armazenamento destes artefatos, é facilitado o acesso e disponibilização dos mesmos aos demais integrantes da equipe.

Durante a execução do processo, o ator pode registrar ocorrências relacionadas às tarefas que estiver executando indicando uma descrição, tempo gasto para realizar o que foi descrito e porcentagem de conclusão da tarefa em questão. Estas ocorrências podem comportar não só a descrição de realização das tarefas, mas também problemas encontrados, decisões tomadas ou mesmo informações relevantes obtidas durante a execução.

Quando um ator deixar de executar uma tarefa, é importante que o mesmo registre o que foi feito e quanto tempo ele gastou, pois estes dados serão importantes na retomada de execução desta tarefa, que pode ser feita pelo mesmo ator ou por outro integrante da equipe. Conhecendo o que já foi feito na tarefa qualquer ator poderá dar continuidade a sua execução, já que este registro servirá para facilitar a comunicação entre os integrantes. Ao disponibilizar um relatório de horas trabalhadas por ator, o mecanismo também permite que o gerente possa acompanhar o tempo dedicado por cada ator na execução do projeto.

O registro das horas trabalhadas e porcentagem de conclusão em cada tarefa também serão úteis ao gerente do projeto por permitir que o mesmo confira se as estimativas previstas no início do projeto condizem com a realidade e, caso necessário, reveja os prazos estipulados ou até mesmo realoque tarefas para diferentes papéis. Por esse motivo, o mecanismo de

execução também oferece recursos para alteração de algumas informações definidas na instanciação e modelagem, como datas e executores para as tarefas.

Registros de dificuldades e problemas encontrados ajudarão o gerente a compreender e justificar possíveis atrasos no projeto, além de servirem como um histórico para que em futuros projetos estes riscos possam ser considerados.

É importante que os atores sempre estejam cientes da sua agenda. Por esse motivo, para evitar atrasos, é interessante que os mesmos sejam notificados de alguma forma sobre a necessidade de início ou fim de uma tarefa além de atrasos existentes. Desta forma, o mecanismo utiliza o e-mail cadastrado para o ator como meio de comunicação para notificá-lo dos casos citados acima. Esta notificação também pode ser útil para os casos em que o ator tenha esquecido de registrar o início ou fim da tarefa e a notificação o lembrará de registrar estas informações no sistema.

Com esta proposta de execução, todo o ciclo do processo estaria integrado uma vez que a modelagem está integrada a instanciação, e esta se encontra integrada a execução. O mecanismo apresentará aos atores uma agenda com seus compromissos no projeto e permitirá que o mesmo navegue pelas informações modeladas e instanciadas oferecendo subsídios para a execução de suas tarefas. Permitirá que o mesmo possa criar registros de ocorrências nas tarefas que servirão como base de informações para os demais atores e ao gerente do projeto. O mecanismo será acessível por todos os integrantes do modelo do processo e servirá também como um centralizador para os artefatos, facilitando o acesso aos mesmos e armazenando-os de forma organizada. Permitirá ao gerente um acompanhamento mais detalhado dos acontecimentos além de permitir a redefinição de algumas informações, com o objetivo de contornar possíveis problemas e diminuir seus impactos.

4.4.1. UM PROCESSO EXEMPLO: EXECUÇÃO

Conforme definido no início do trabalho, os recursos utilizados para desenvolver estes mecanismos devem ser livres. Por isso, o mecanismo de execução foi desenvolvido como uma aplicação Web utilizando a linguagem e tecnologias Java, banco de dados MySQL e o Apache TomCat como servidor de aplicação.

Para iniciar a execução de um processo, o usuário deve inicialmente se cadastrar para ter acesso ao sistema. Se cadastrando, este usuário terá o perfil de gerente do processo de software que ele iniciar. O início de um processo é feito após o usuário importar o artefato

gerado pelo mecanismo de instanciação, além de dar um nome e uma breve descrição para o projeto.

Ao importar os dados do modelo, o mecanismo de execução processa o XML e armazena as informações necessárias para o controle e acompanhamento do processo. O mecanismo identifica os atores do processo, cria um usuário no sistema para cada e envia um e-mail notificando sua conta e senha de acesso.

Após importar os dados, o gerente passa a poder gerar relatórios de horas trabalhadas por cada ator e pode ver o gráfico que indica o andamento das tarefas. Esta visão é dada por meio de um Gráfico de Gantt, que permite uma visão de cronograma conforme apresentado na FIG 4.11.

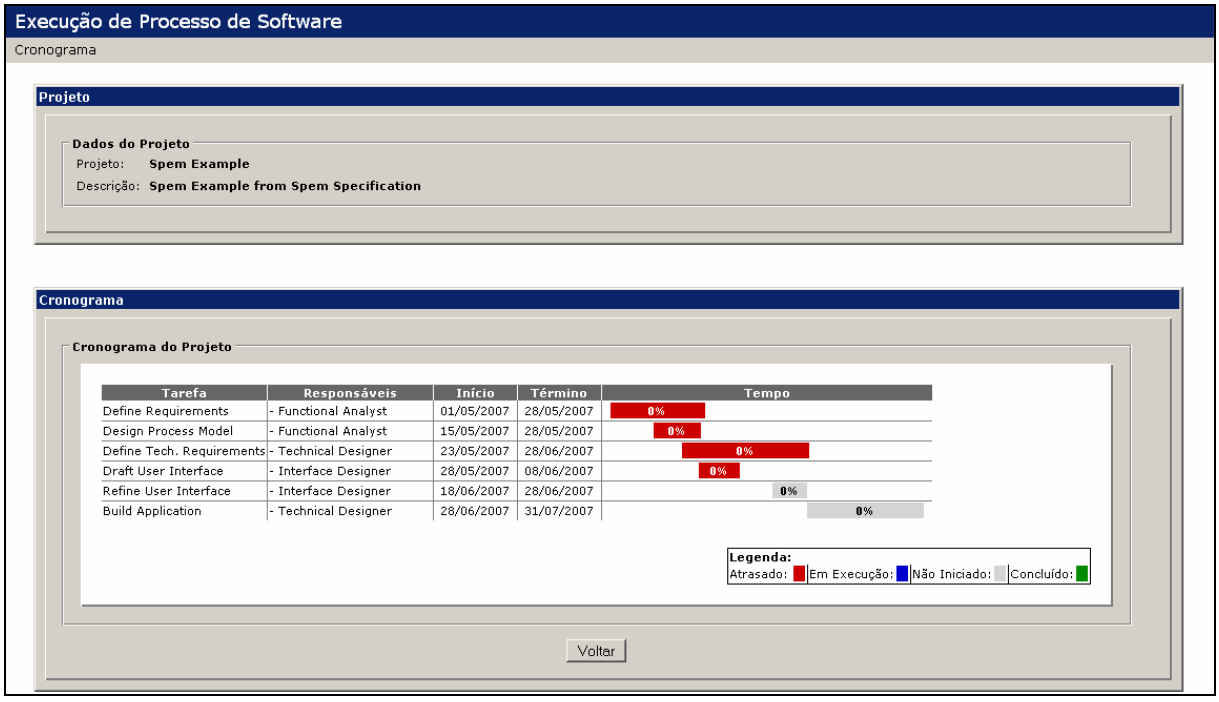


FIG 4.11 – Tela que exibe o gráfico de andamento do projeto.

Os atores do sistema podem ver este mesmo gráfico do projeto, além de terem a opção de ver esta tela filtrada apenas com as tarefas relacionadas aos seus papéis. Esta tela funciona como a agenda do ator, exibindo as tarefas que ele deve realizar e o prazo para cada uma delas.

Selecionando uma tarefa da lista, é exibida uma tela de detalhes da tarefa onde o usuário pode ver um conjunto de dados relacionados a ela. Nesta mesma tela o usuário pode registrar as ocorrências e ver quais são os artefatos de entrada e saída.

Na FIG 4.12 é exibida a tela de detalhes da tarefa no estado de inclusão de novos registros. Nesta mesma tela é possível alternar para visualizar os artefatos ou visualizar os passos da tarefa.

Assim como a tarefa, o artefato também possui uma tela de detalhes que é responsável por exibir seus dados e permite o envio ou a recuperação de uma versão do documento. Esta tela é apresentada na FIG 4.13.

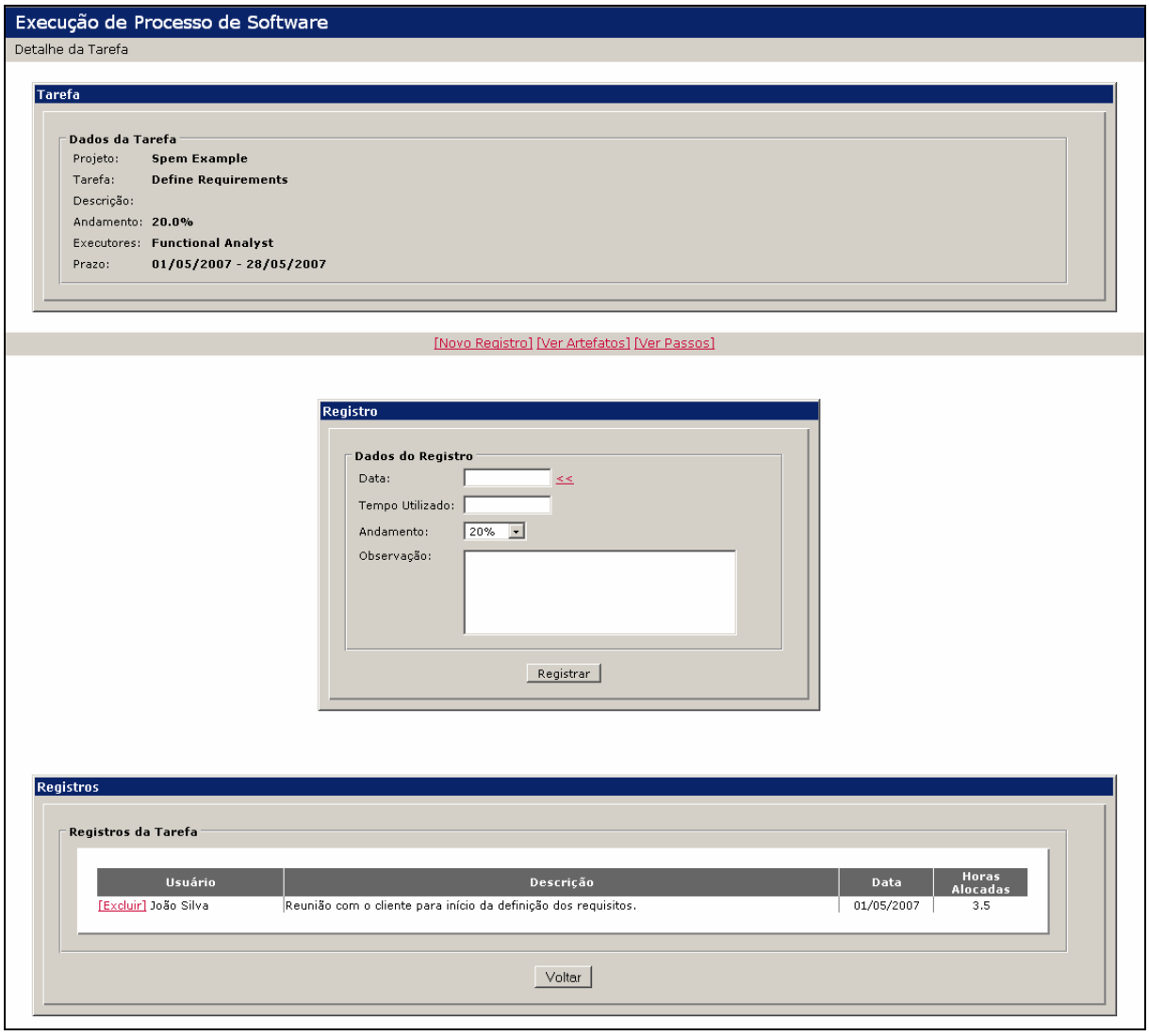


FIG 4.12 – Tela de detalhes da tarefa.

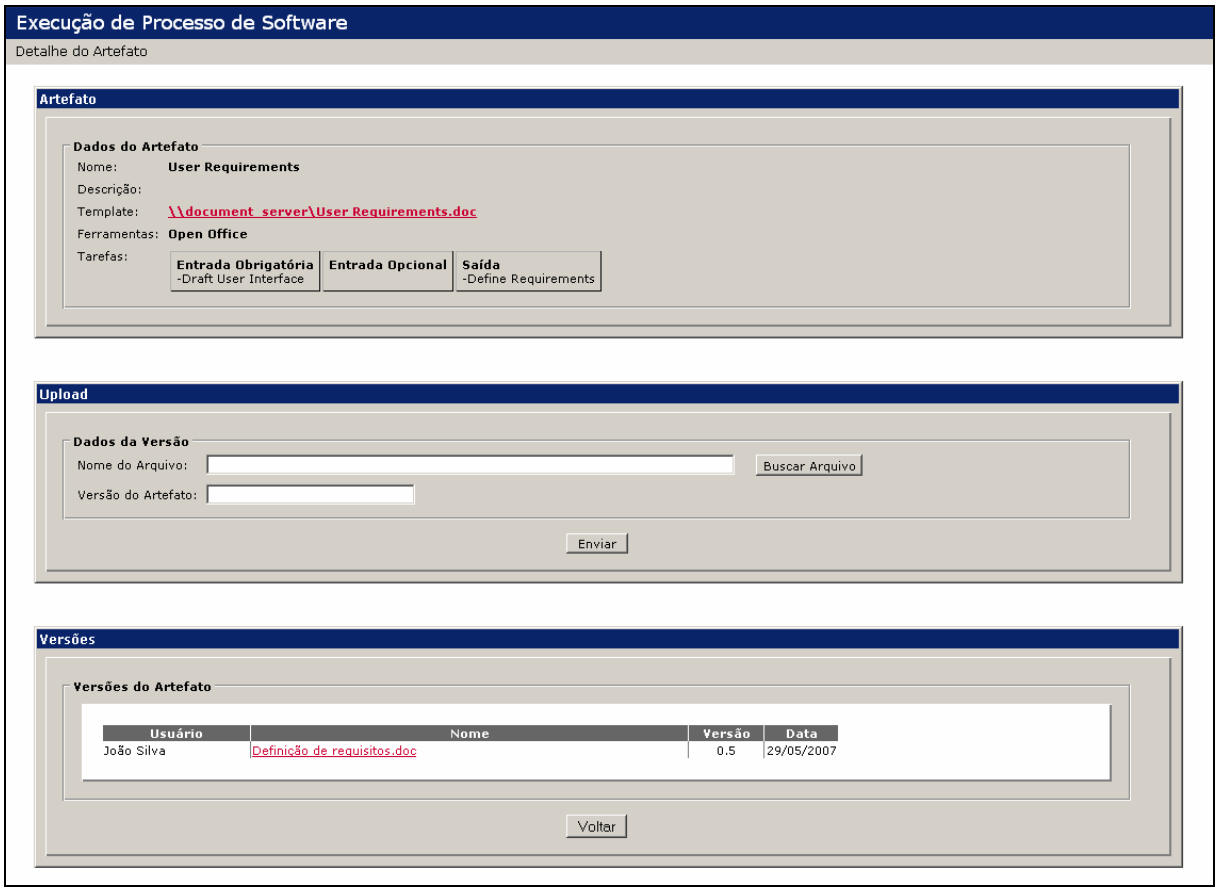


FIG 4.13 – Tela de detalhes do artefato.

Analisando as ocorrências e artefatos gerados ao longo da execução do processo, o gerente tem a possibilidade de alterar os executores ou mesmo alterar as datas de início e fim das tarefas.

Para manter os atores cientes das tarefas a serem executadas e ajudar para que o estado das tarefas esteja sempre atualizado, o sistema envia diariamente um e-mail para os atores indicando as tarefas que deverão iniciar ou finalizar no dia em questão, e também indicando as tarefas que já deveriam ter iniciado ou finalizado, indicando que as mesmas estão atrasadas.

4.5. CONSIDERAÇÕES SOBRE O AIPS

De acordo com o que foi apresentado, percebe-se que a proposta atinge os principais objetivos traçados no início do trabalho. A proposta é suportada por mecanismos criados sob a política de Software Livre, que trabalham de forma integrada nas fases de modelagem, instanciação e execução de processos de software, oferecendo na fase de execução a

possibilidade de um acompanhamento detalhado por parte do gerente e dos demais atores. O mecanismo permite ainda que um mesmo modelo de processos seja instanciado diversas vezes e oferece a possibilidade de redefinição de algumas relações criadas na modelagem ou instanciação, durante a execução do processo.

Desta forma foi possível atingir as principais fases do ciclo de vida de um processo de software permitindo que a definição deste processo seja feita utilizando conceitos interessantes existentes na ferramenta de modelagem EPF; permitindo que na instanciação seja possível alocar recursos, definir prazos para as tarefas e apontar ferramentas para geração de artefatos e permitindo que na execução o ator tenha acesso a todas as informações necessárias para que possa realizar seu trabalho, registrar o andamento de suas tarefas, além de permitir ao gerente acompanhar todas estas informações.

A integração é alcançada com a utilização dos produtos gerados ao fim das duas primeiras fases (modelagem e instanciação) como fonte de dados para a fase seguinte. Assim como o produto gerado na exportação do modelo do processo para um arquivo XML pelo EPF serve de fonte de informações para o mecanismo de instanciação, o produto gerado por este mecanismo alimenta o mecanismo criado para a execução sem a necessidade de qualquer adaptação neste produto. Como o produto gerado na modelagem servirá apenas como fonte de dados inicial para a fase de instanciação, este mesmo produto, gerado uma única vez, poderá servir como fonte de dados também para outras instâncias do mesmo processo.

A proposta de instanciação compreende três importantes passos, onde o primeiro consiste na definição de atores que irão assumir os papéis existentes no processo e com isso será possível criar a agenda de tarefas de cada integrante da equipe uma vez que no modelo abstrato as tarefas já estão alocadas a estes papéis. O segundo passo consiste na definição de prazos para a realização das tarefas onde cabe ao gerente informar a data de início e fim de cada uma destas tarefas, além de poder definir iterações de um determinado conjunto de tarefas. O terceiro passo consiste na indicação de ferramentas que deverão ser utilizadas para gerar determinado artefato, podendo ainda apontar para um documento de *template* para este artefato.

A proposta de execução permite aos atores conhecerem detalhes importantes relacionados às tarefas e artefatos definidos nas fases de modelagem e instanciação e possibilita o registro de informações relacionadas às suas atividades no projeto. Estas informações serão úteis para os demais atores e também aos gerentes que, analisando estes dados, poderão optar por algum tipo de alteração no processo objetivando solucionar possíveis problemas.

5. PROVA DE CONCEITO

Esse capítulo descreve o uso prático do modelo proposto em situações que envolvem a modelagem, instanciação e execução de processos de software. Estas provas de conceito foram realizadas com o objetivo de avaliar a ferramenta de modelagem proposta, o mecanismo de instanciação criado, seus recursos e o mecanismo de execução.

A primeira prova de conceito foi feita sobre *Open Unified Process – OpenUP* (BALDUINO, 2006), que é um processo disponibilizado junto com o EPF. O modelo foi exportado e o produto desta exportação submetido ao mecanismo de instanciação. Alguns dados fictícios foram definidos na instanciação para que o modelo pudesse ser executado.

A segunda foi feita sobre o Método de Desenvolvimento de Sistemas (PRODERJ, 2006) que é um processo criado e utilizado pelo Centro de Tecnologia da Informação e Comunicação do Estado do Rio de Janeiro – PRODERJ. A partir do documento de especificação deste processo, este foi modelado no EPF para avaliação da utilização da ferramenta. Em seguida, o modelo foi exportado para que pudesse ser feita a avaliação da instanciação deste modelo. Novamente, alguns dados fictícios foram definidos na instanciação para gerar um modelo executável do processo.

Em ambos os casos, os processos foram executados para avaliação e validação do mecanismo. Foi possível avaliar os recursos disponibilizados e algumas situações foram criadas para que o comportamento do mecanismo também fosse avaliado nos diversos estados possíveis para o processo.

5.1. OPEN UNIFIED PROCESS

O *Open Unified Process* (*OpenUP*) é uma versão código aberto do *Rational Unified Process* (RUP) desenvolvida por colaboradores do *Eclipse Process Framework*. O *OpenUP/Basic* é a versão mais ágil e leve do *OpenUP* e é o processo que será utilizado neste estudo de caso, por ser disponibilizado juntamente com o EPF. Ele é um processo iterativo, que possui iterações ao longo de suas quatro fases: concepção, elaboração, construção e transição.

O *OpenUP/Basic* é um processo de desenvolvimento de software elaborado para ser mínimo, completo e extensível. Mínimo por possuir apenas conteúdo fundamental, completo por conter tudo que é necessário para se construir um sistema, e extensível por poder ser utilizado como base, onde qualquer elemento do processo pode ser adicionado ou retirado

conforme a necessidade. Ele disponibiliza exemplos de WBS para cada iteração e um exemplo de WBS para um processo completo composto pelas suas quatro fases.

5.1.1. MODELAGEM

A modelagem de um processo deve ser capaz de representar todos os elementos existentes neste processo. Nesta modelagem o processo deve estar suficientemente detalhado de forma que, com estas informações, seja possível controlar a execução do processo.

A FIG 5.1 exibe o diagrama de atividades que descreve o processo como um todo e a ordem de suas quatro fases de iteração.

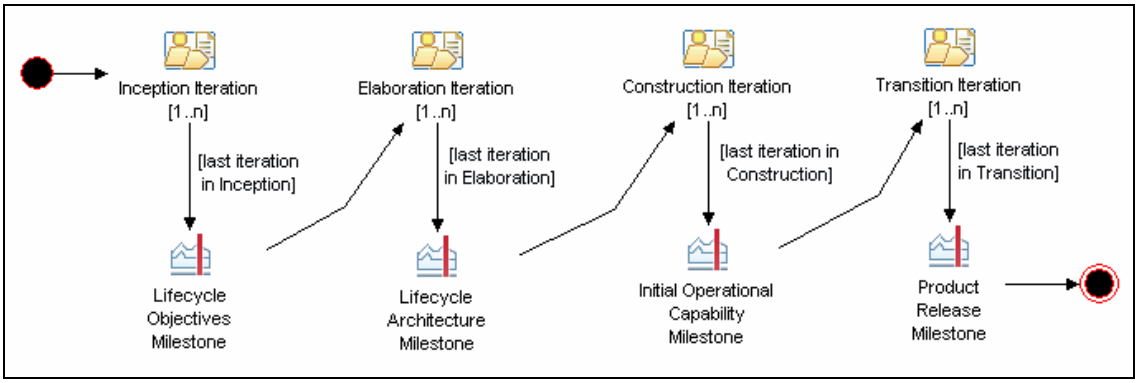


FIG 5.1 – Diagrama de Atividades do processo.

A primeira fase do processo é a fase de concepção. Esta fase é composta por um conjunto de cinco atividades que se organizam conforme ilustrado na FIG 5.2. As atividades que a compõem são:

- *Initiate Project*: Define o escopo do projeto e a sua duração, baseado na priorização dos requisitos; identifica os riscos, determina como serão tratados e cria o plano de projetos.
- *Manage Iteration*: Inicia a iteração e distribui tarefas aos componentes da equipe; monitora e comunica o estado do projeto aos clientes externos; identifica e trata os problemas.
- *Manage Requirement*: Detalha um conjunto de requisitos (um ou mais casos de uso, cenários, entre outros).
- *Determine Architectural Feasibility*: Confirma a possibilidade de realização do projeto com a construção de uma prova de conceito arquitetural.

- *Assess and Plan Iteration*: Avalia a iteração atual e utiliza as lições aprendidas para planejar a próxima iteração.

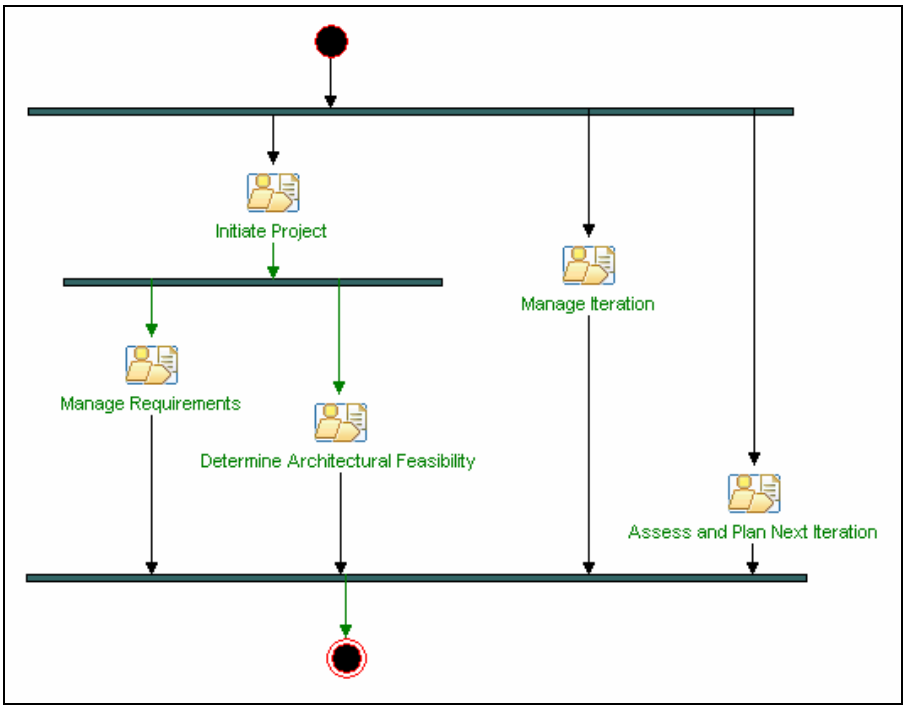


FIG 5.2 – Diagrama de Atividades da fase concepção.

A segunda fase do processo é a fase de elaboração, que é composta por um outro conjunto de atividades e é apresentada na FIG 5.3. Neste conjunto de atividades se encontram algumas que já foram realizadas anteriormente e outras quatro que ainda não haviam aparecido no processo, que são:

- *Define the Architecture*: Define uma arquitetura candidata para o sistema, baseado em experiências obtidas em sistemas ou domínios de problemas semelhantes.
- *Develop Solution (for requirement) (within context)*: Projeta, implementa, testa e integra a solução para um requisito em um determinado contexto.
- *Validate Build*: Testa e avalia os requisitos desenvolvidos, da perspectiva do sistema.
- *Ongoing Tasks*: Executa as tarefas que não necessariamente fazem parte do cronograma do projeto.

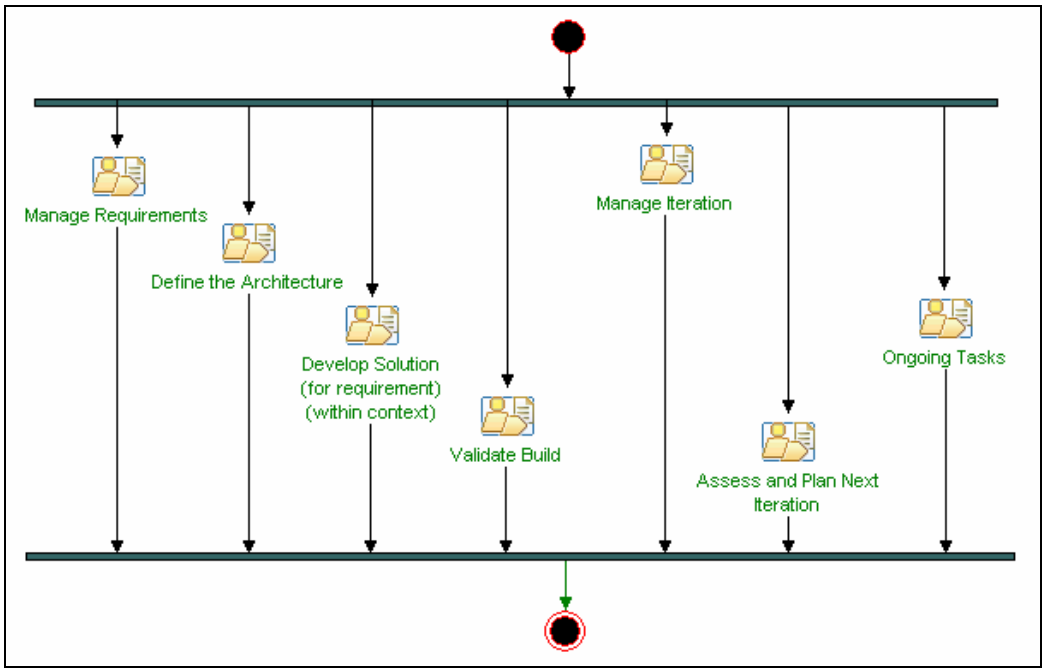


FIG 5.3 – Diagrama de Atividades da fase elaboração.

A fase seguinte do processo é a fase de construção, que possui a estrutura de atividades bastante semelhante com a fase anterior, havendo apenas a substituição da atividade "*Define the Architeture*" pela atividade "*Refine the Architeture*" (FIG 5.4). Esta atividade é responsável por tomar decisões concretas sobre a arquitetura, para fornecer uma direção ao trabalho de desenvolvimento na iteração.

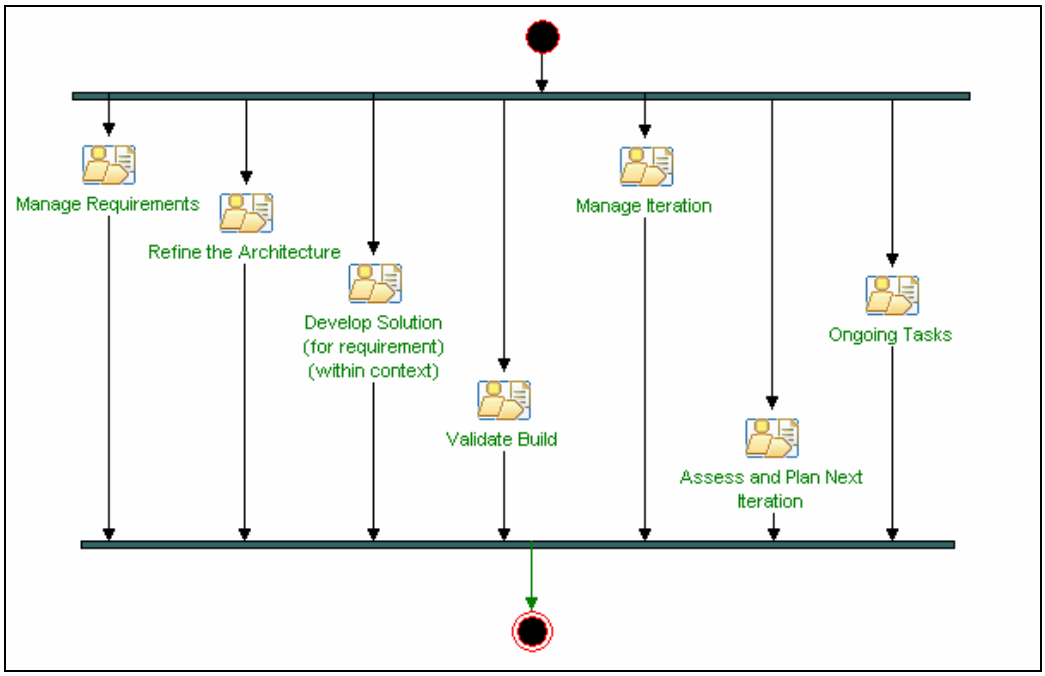


FIG 5.4 – Diagrama de Atividades da fase construção.

Por último, é realizada a fase de transição, que sincroniza a finalização de quatro atividades para avaliar e finalizar o projeto. Esta atividade, "*Assess and Close-out Project*", avalia a iteração final e obtém a opinião do cliente; conduz uma retrospectiva do projeto e o finaliza.

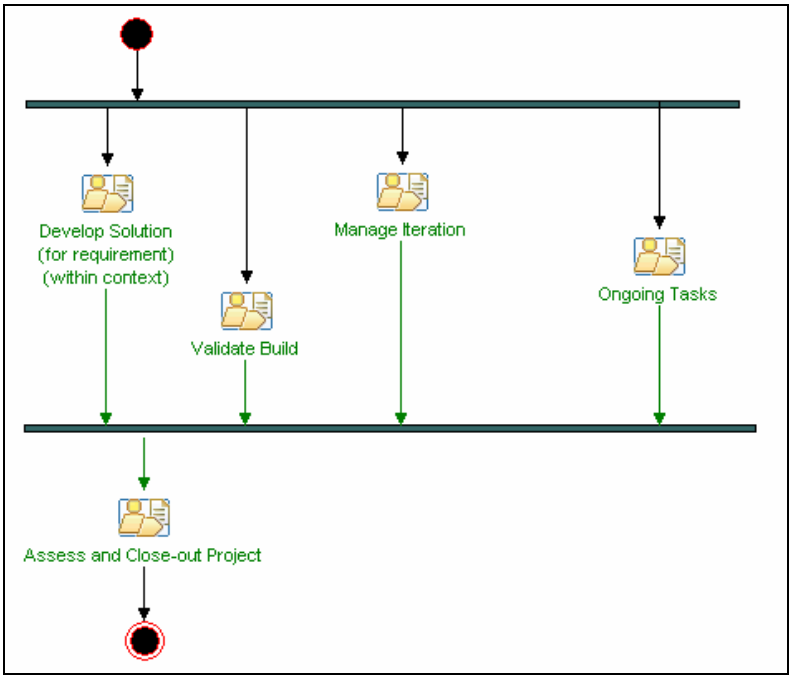


FIG 5.5 – Diagrama de Atividades da fase de transição.

Cada uma destas atividades é composta por um conjunto de tarefas e , assim como todos os elementos presentes neste processo, todas elas foram modeladas dispersas por pacotes do *Method Content* que comporta o conteúdo reutilizável do processo. Foram ainda categorizadas em seis grupos: Análise e Projeto, Gerência de Configuração e Mudanças, Implementação, Gerência do Projeto, Requisitos e Testes. Estas tarefas, divididas em seus respectivos grupos, podem ser visualizadas na FIG 5.6.

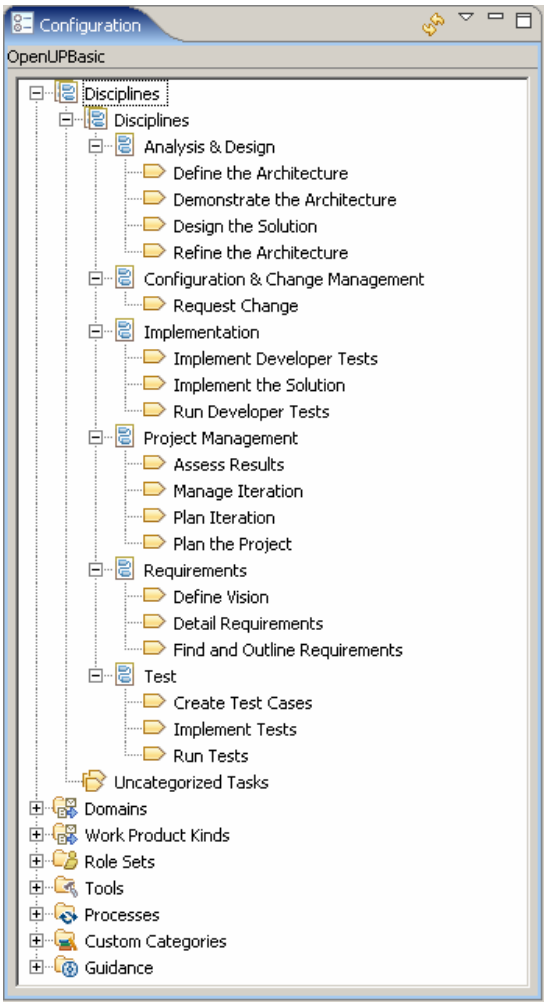


FIG 5.6 – Lista de tarefas organizadas por grupos.

No cadastro da tarefa, além do formulário de cadastro da descrição do elemento (FIG 5.7), também foram definidos os papéis que a realizarão (FIG 5.8), os passos para sua realização (FIG 5.9) e os artefatos de entrada e saída (FIG 5.10).

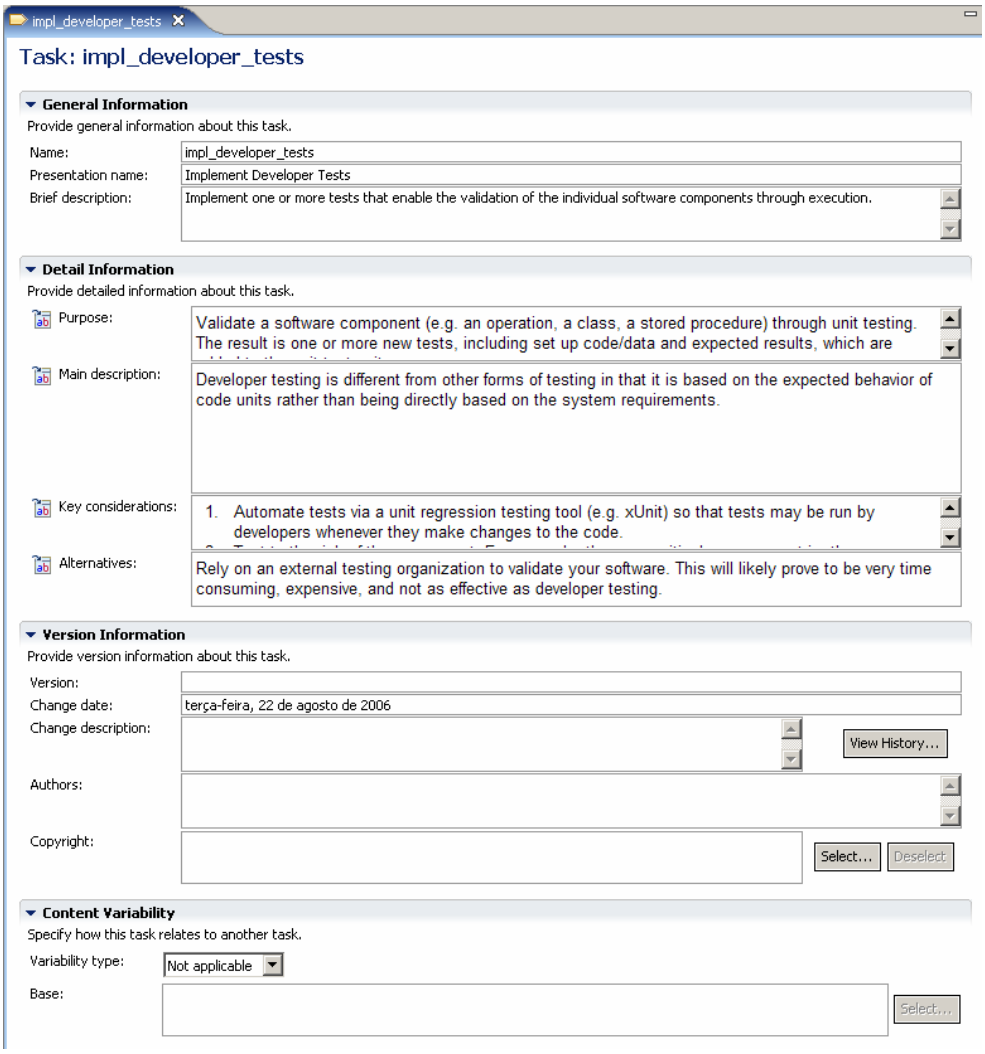


FIG 5.7 – Tela para definição de uma tarefa.

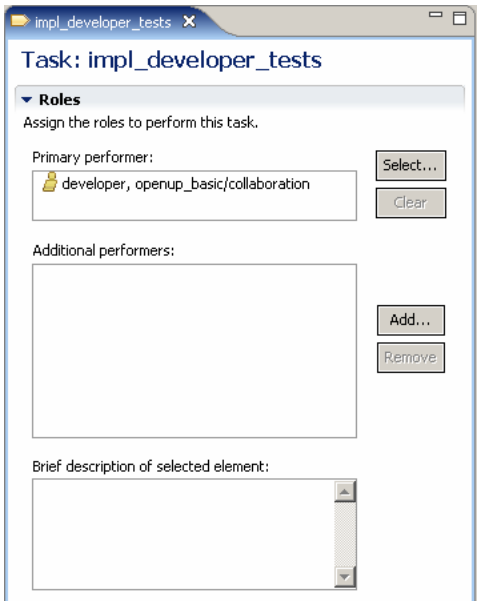


FIG 5.8 – Tela para definição dos executores da tarefa.

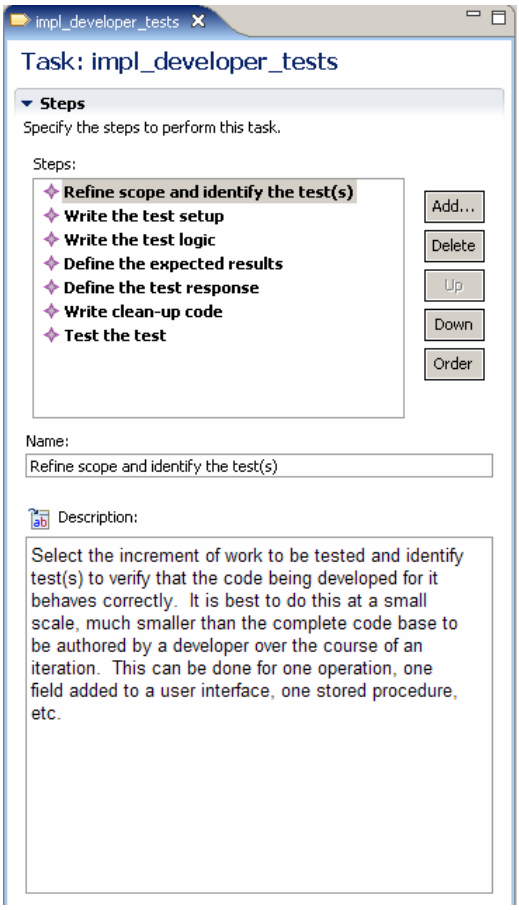


FIG 5.9 – Tela para definição de passos da tarefa.

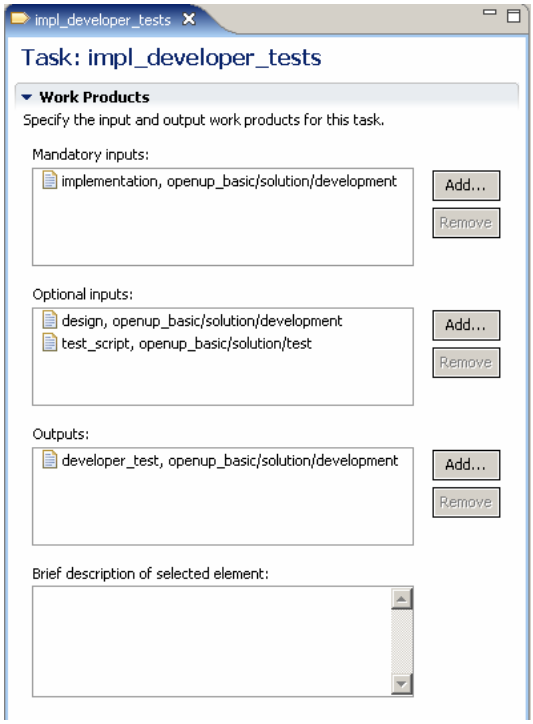


FIG 5.10 – Tela para definição dos artefatos de entrada e saída da tarefa.

Após a modelagem dos elementos, foi feita a modelagem das atividades como pequenos processos, onde os elementos do *Method Content* são referenciados. Estes pequenos processos serão depois utilizados para compor o processo completo de desenvolvimento e foram criados na pasta *Capability Patterns* conforme apresentado na FIG 5.11.

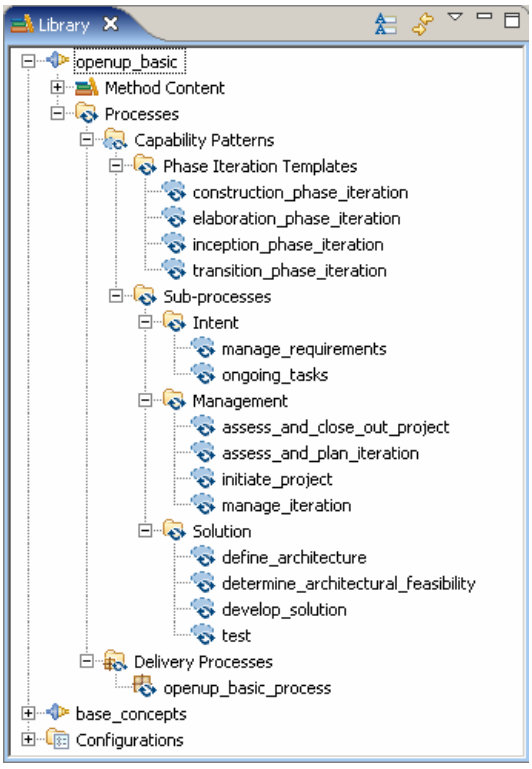
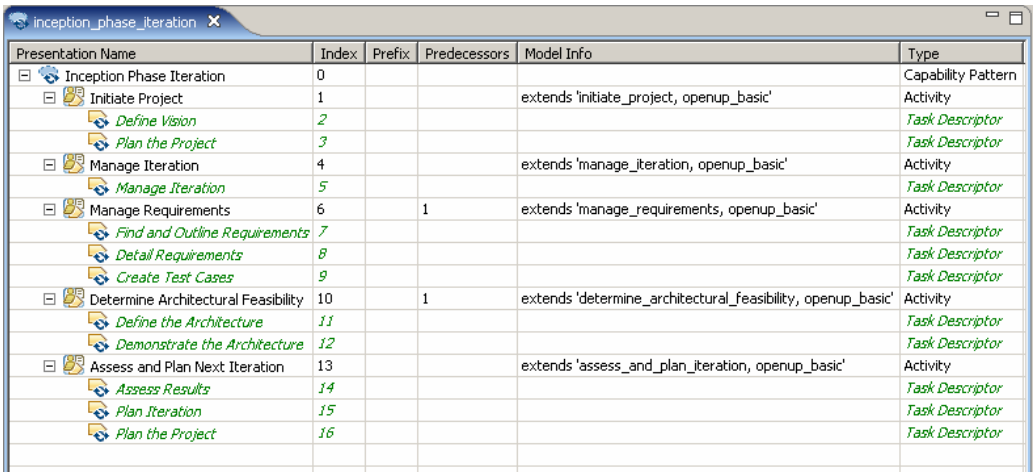


FIG 5.11 – Lista de processos na pasta *Capability Patterns*.

Dentro da pasta *Capability Patterns*, existem outras duas pastas: uma para os processos que representam as fases de iteração e outra para as atividades. Enquanto estas atividades (ou sub-processos) referenciam os elementos modelados no conteúdo reutilizável, os processos da fase de iteração são compostos por atividades, que estendem estes sub-processos. O processo de Concepção é apresentado na FIG 5.12, onde é possível observar também o conteúdo dos sub-processos.



inception_phase_iteration

Presentation Name	Index	Prefix	Predecessors	Model Info	Type
Inception Phase Iteration	0				Capability Pattern
Initiate Project	1			extends 'initiate_project, openup_basic'	Activity
Define Vision	*2*				*Task Descriptor*
Plan the Project	*3*				*Task Descriptor*
Manage Iteration	4			extends 'manage_iteration, openup_basic'	Activity
Manage Iteration	*5*				*Task Descriptor*
Manage Requirements	6		1	extends 'manage_requirements, openup_basic'	Activity
Find and Outline Requirements	*7*				*Task Descriptor*
Detail Requirements	*8*				*Task Descriptor*
Create Test Cases	*9*				*Task Descriptor*
Determine Architectural Feasibility	10		1	extends 'determine_architectural_feasibility, openup_basic'	Activity
Define the Architecture	*11*				*Task Descriptor*
Demonstrate the Architecture	*12*				*Task Descriptor*
Assess and Plan Next Iteration	13			extends 'assess_and_plan_iteration, openup_basic'	Activity
Assess Results	*14*				*Task Descriptor*
Plan Iteration	*15*				*Task Descriptor*
Plan the Project	*16*				*Task Descriptor*

FIG 5.12 – Processo da fase de Concepção.

Todas as tarefas são executadas por um ou mais papéis, e estes também devem estar modelados no processo. Os principais papéis são:

- ***Analyst***: Representa interesses do cliente e do usuário final, recolhendo informações das partes interessadas para compreender o problema a ser resolvido, capturando e priorizando os requisitos.
- ***Any Role***: Desempenhado por qualquer integrante da equipe, a fim executar tarefas gerais.
- ***Architect***: Responsável pela arquitetura do sistema, incluindo decisões técnicas.
- ***Developer***: Responsável por desenvolver parte do sistema, incluindo projetá-lo para que se encaixe na arquitetura.
- ***Project Manager***: Conduz ao planejamento do projeto, coordena interações com as partes interessadas, e mantém a equipe de projeto focada.
- ***Stakeholder***: Representa os grupos de interesse cujas necessidades devem ser satisfeitas pelo projeto. É um papel que pode ser desempenhado por qualquer um que seja afetado pelo resultado do projeto.
- ***Tester***: Responsável pelas atividades de realização dos testes, que envolve identificar, definir, executar e conduzir os testes necessários, assim como registrar e analisar os resultados.

No *Method Content*, os elementos encontram-se organizados em pacotes, onde cada pacote possui seus papéis, suas tarefas e seus artefatos correspondentes. Na FIG 5.13 observa-se que um conjunto de papéis pertencentes ao processo foi modelado no pacote

"*collaboration*", já na FIG 5.14, é possível identificar o conjunto completo de pacotes e sub-pacotes criados para comportar todos os elementos do processo.

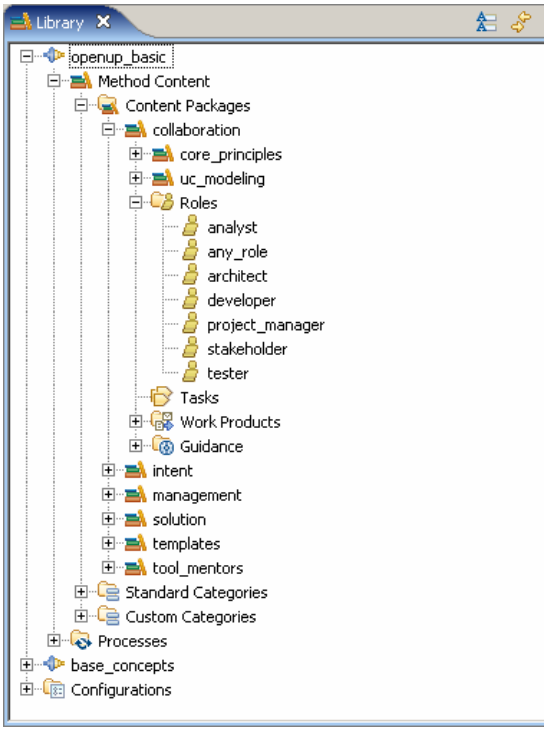


FIG 5.13 – Lista de papéis do pacote "*collaboration*".

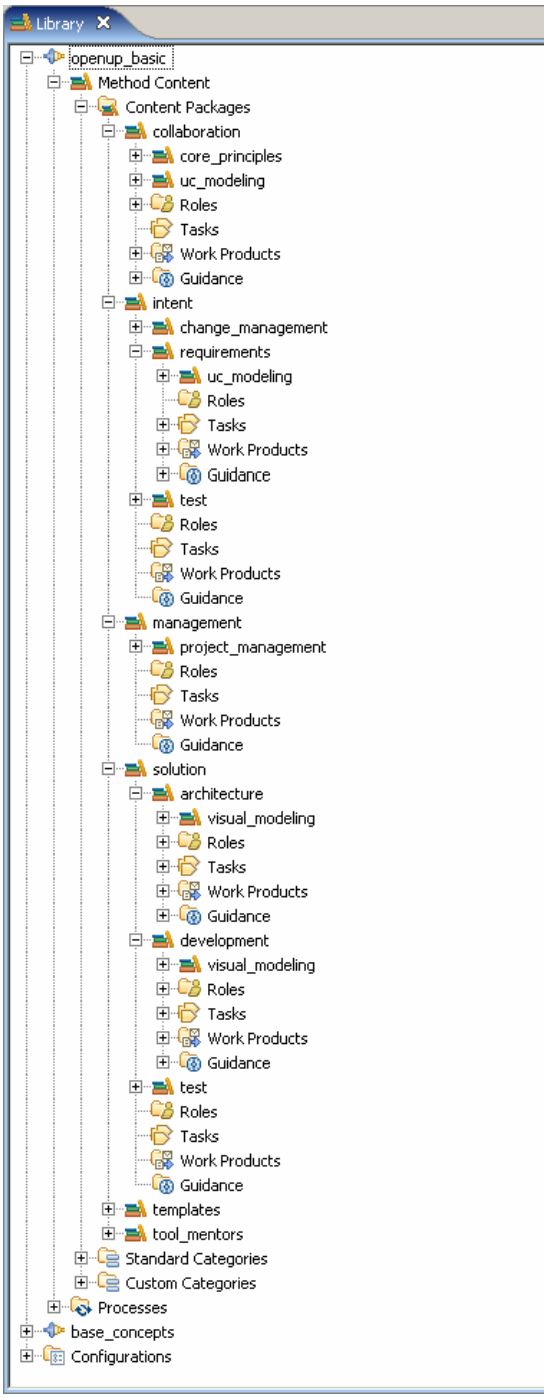


FIG 5.14 – Lista pacotes modelados.

Todo o processo é composto por estes sete papéis existentes no pacote "*collaboration*", mais outros seis papéis que são variações destes sete. Para cada um destes papéis estão definidos seu nome e uma breve descrição sobre ele, mas há ainda um amplo conjunto de informações adicionais que podem ser definidas. Na FIG 5.15, é apresentada como exemplo a tela de definição do papel "Stakeholder".

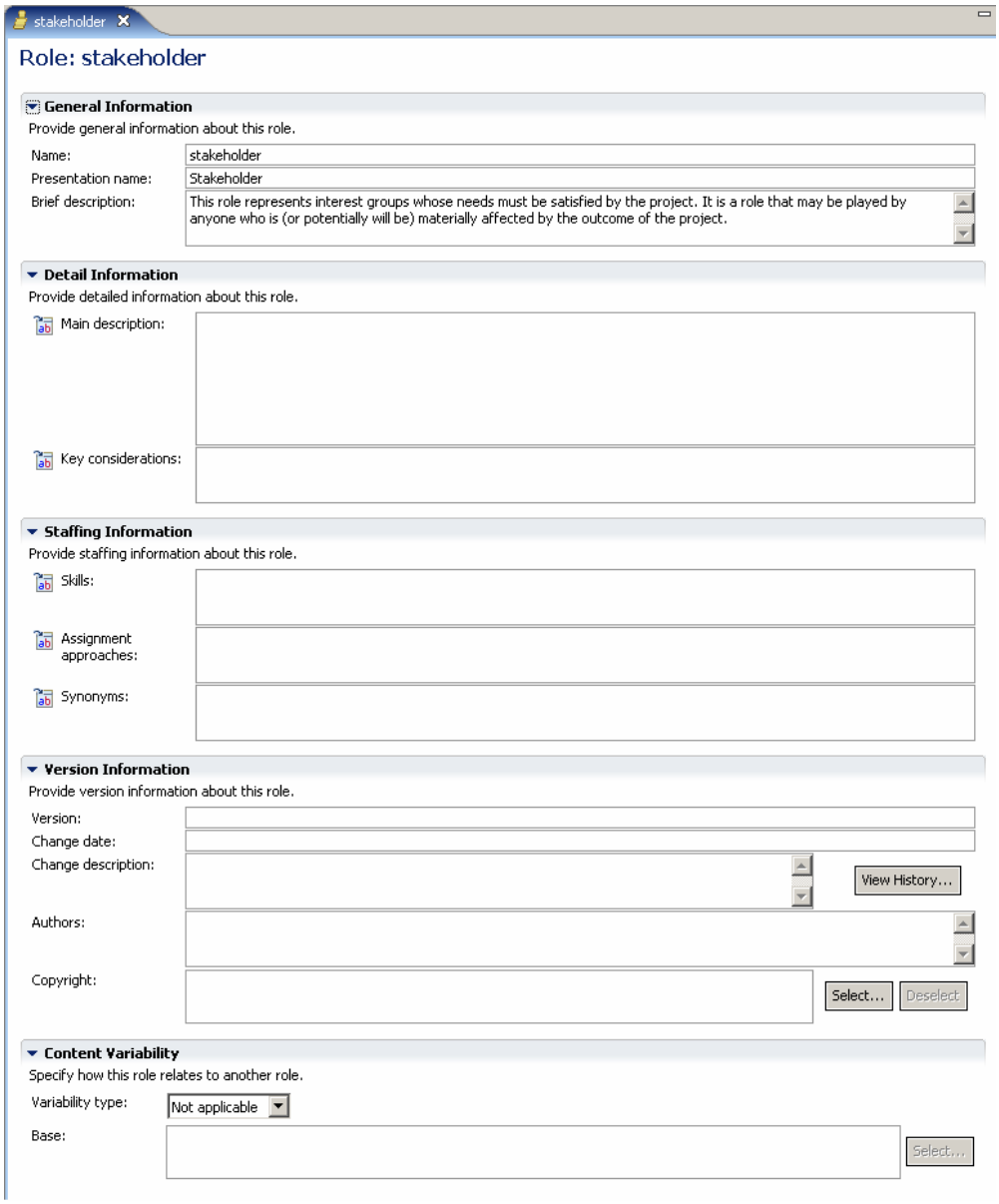


FIG 5.15 – Tela para definição de um papel.

Os artefatos do processo também foram modelados dispersos pelos pacotes existentes no *Method Content* e categorizados em seis grupos: Arquitetura, Gerência de Configuração, Desenvolvimento, Gerência de Projetos, Requisitos e Testes (FIG 5.16). O cadastro de artefatos no EPF também possui um formulário extenso e detalhado, que é apresentado na FIG 5.17.

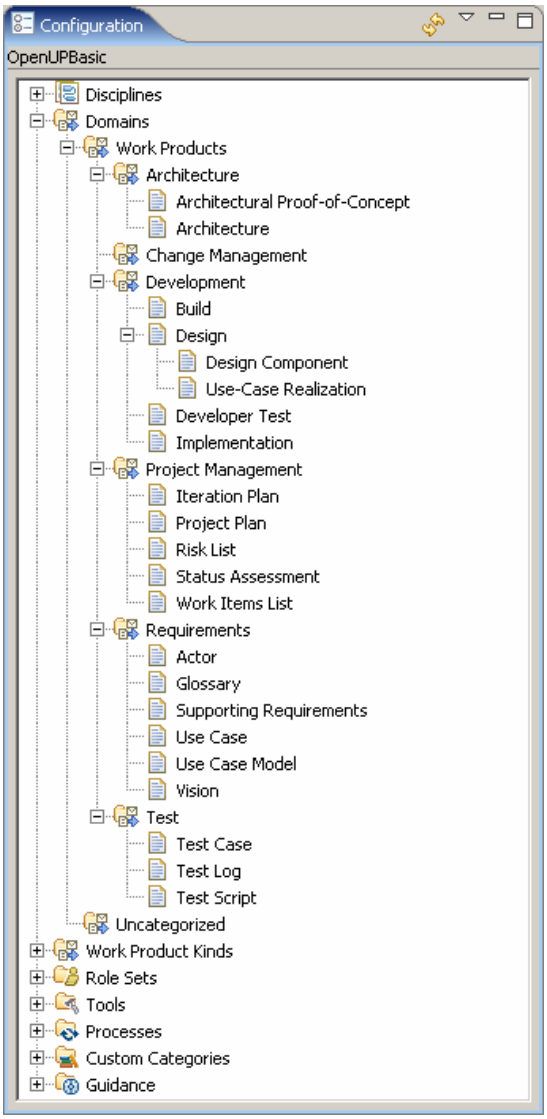


FIG 5.16 – Lista de artefatos organizados por grupos.

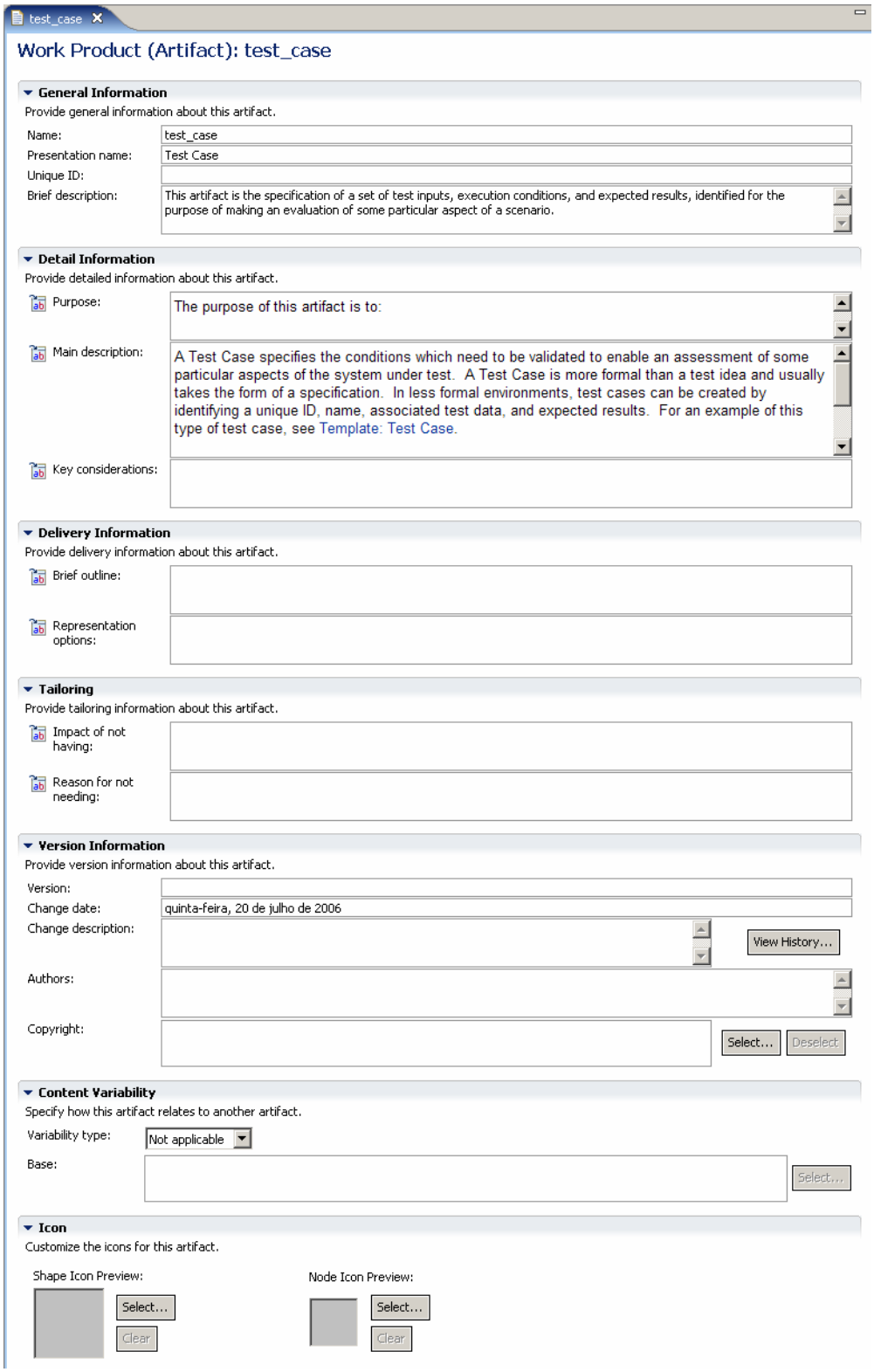
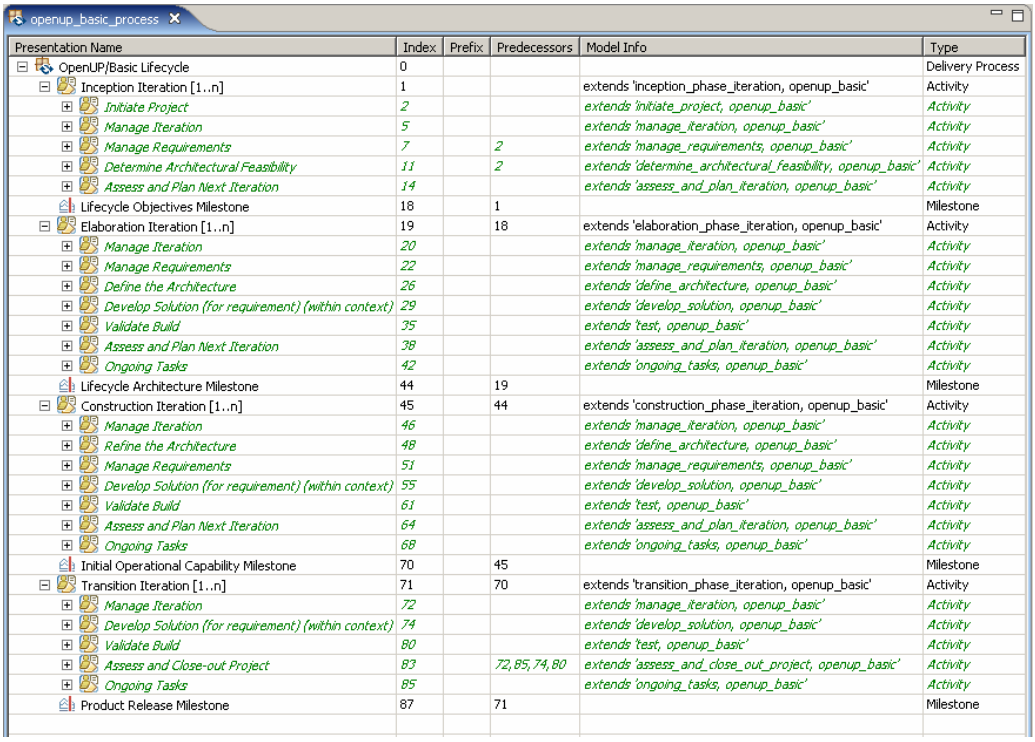


FIG 5.17 – Tela para definição de um artefato.

O WBS do processo completo é exibido na FIG 5.18.



openup_basic_process

Presentation Name	Index	Prefix	Predecessors	Model Info	Type
OpenUP/Basic Lifecycle	0				Delivery Process
Inception Iteration [1..n]	1			extends 'inception_phase_iteration, openup_basic'	Activity
Initiate Project	*2*			*extends 'initiate_project, openup_basic'*	*Activity*
Manage Iteration	*5*			*extends 'manage_iteration, openup_basic'*	*Activity*
Manage Requirements	*7*		*2*	*extends 'manage_requirements, openup_basic'*	*Activity*
Determine Architectural Feasibility	*11*		*2*	*extends 'determine_architectural_feasibility, openup_basic'*	*Activity*
Assess and Plan Next Iteration	*14*			*extends 'assess_and_plan_iteration, openup_basic'*	*Activity*
Lifecycle Objectives Milestone	18		1		Milestone
Elaboration Iteration [1..n]	19		18	extends 'elaboration_phase_iteration, openup_basic'	Activity
Manage Iteration	*20*			*extends 'manage_iteration, openup_basic'*	*Activity*
Manage Requirements	*22*			*extends 'manage_requirements, openup_basic'*	*Activity*
Define the Architecture	*26*			*extends 'define_architecture, openup_basic'*	*Activity*
Develop Solution (for requirement) (within context)	*29*			*extends 'develop_solution, openup_basic'*	*Activity*
Validate Build	*35*			*extends 'test, openup_basic'*	*Activity*
Assess and Plan Next Iteration	*38*			*extends 'assess_and_plan_iteration, openup_basic'*	*Activity*
Ongoing Tasks	*42*			*extends 'ongoing_tasks, openup_basic'*	*Activity*
Lifecycle Architecture Milestone	44		19		Milestone
Construction Iteration [1..n]	45		44	extends 'construction_phase_iteration, openup_basic'	Activity
Manage Iteration	*46*			*extends 'manage_iteration, openup_basic'*	*Activity*
Refine the Architecture	*48*			*extends 'define_architecture, openup_basic'*	*Activity*
Manage Requirements	*51*			*extends 'manage_requirements, openup_basic'*	*Activity*
Develop Solution (for requirement) (within context)	*55*			*extends 'develop_solution, openup_basic'*	*Activity*
Validate Build	*61*			*extends 'test, openup_basic'*	*Activity*
Assess and Plan Next Iteration	*64*			*extends 'assess_and_plan_iteration, openup_basic'*	*Activity*
Ongoing Tasks	*68*			*extends 'ongoing_tasks, openup_basic'*	*Activity*
Initial Operational Capability Milestone	70		45		Milestone
Transition Iteration [1..n]	71		70	extends 'transition_phase_iteration, openup_basic'	Activity
Manage Iteration	*72*			*extends 'manage_iteration, openup_basic'*	*Activity*
Develop Solution (for requirement) (within context)	*74*			*extends 'develop_solution, openup_basic'*	*Activity*
Validate Build	*80*			*extends 'test, openup_basic'*	*Activity*
Assess and Close-out Project	*83*		*72,85,74,80*	*extends 'assess_and_close_out_project, openup_basic'*	*Activity*
Ongoing Tasks	*85*			*extends 'ongoing_tasks, openup_basic'*	*Activity*
Product Release Milestone	87		71		Milestone

FIG 5.18 – WBS do *OpenUP/Basic*.

Este processo, quando exportado, gera um arquivo em XML que conterá todos os elementos presentes no modelo, bem como todos os seus processos e sub-processos. O trecho deste XML que representa uma parte da especificação do processo completo é exibido a seguir, e é importante destacar que ele apenas referencia as fases de iteração. Estas fases de iteração, por sua vez também irão referenciar os sub-processos, que irão referenciar os conteúdos reutilizáveis.

```
<MethodPackage xsi:type="uma:ProcessComponent" name="openup_basic_process" ... >
  <Process xsi:type="uma:DeliveryProcess" name="openup_basic_process" ... >
  ...
    <BreakdownElement xsi:type="uma:Activity" name="inception_phase_iteration" ... >
      <SuperActivity>_0uyGoMlgEdmt3adZL5Dmdw</SuperActivity>
      <Concept>_0hmKgBOMEduCNqgZdt_OaA</Concept>
    </BreakdownElement>
  ...
    <BreakdownElement xsi:type="uma:Activity" name="elaboration_phase_iteration" ... >
      <SuperActivity>_0uyGoMlgEdmt3adZL5Dmdw</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_8bf3QBOSEduCNqgZdt_OaA">
          _y1uwgBOKEduCNqgZdt_OaA
      </Predecessor>
      <Concept>_2plxwBOMEduCNqgZdt_OaA</Concept>
    </BreakdownElement>
  ...
    <BreakdownElement xsi:type="uma:Activity" name="construction_phase_iteration" ... >
      <SuperActivity>_0uyGoMlgEdmt3adZL5Dmdw</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_88coMBOSEduCNqgZdt_OaA">
          _16nd0BOKEduCNqgZdt_OaA
      </Predecessor>
      <Concept>_48EKsBOMEduCNqgZdt_OaA</Concept>
    </BreakdownElement>
  ...
    <BreakdownElement xsi:type="uma:Activity" name="transition_phase_iteration" ... >
      <SuperActivity>_0uyGoMlgEdmt3adZL5Dmdw</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_9abksBOSEduCNqgZdt_OaA">
          _31E_YBOKEduCNqgZdt_OaA
      </Predecessor>
      <Concept>__ca5UBOMEduCNqgZdt_OaA</Concept>
    </BreakdownElement>
  ...
  </Process>
</MethodPackage>
```

5.1.2. INSTANCIAÇÃO

A instanciação é a fase em que o modelo abstrato do processo é estendido para comportar o conjunto de informações necessárias para a execução de uma instância do modelo do processo. Desta forma, o modelo abstrato do processo se transformará em um modelo executável, o que possibilitará o acompanhamento detalhado das tarefas existentes no projeto.

Então, será utilizado o produto gerado pela ferramenta de modelagem como fonte de dados para o mecanismo de instanciação. O XML deve ser importado, conforme apresentado na FIG 5.19, para que o mecanismo carregue as informações do processo e permita que novas informações possam ser associadas.

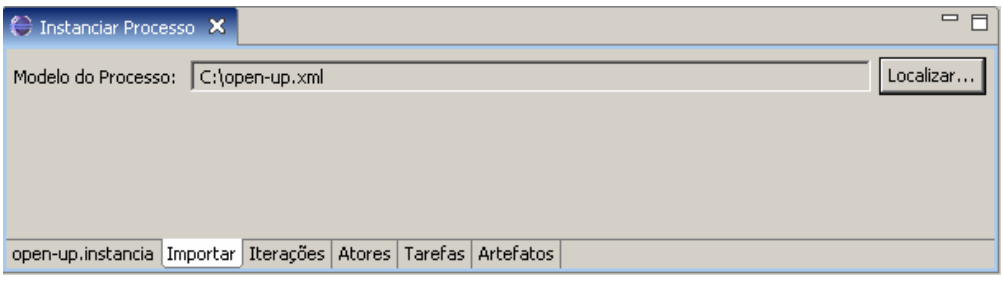


FIG 5.19 – Tela para importação do modelo abstrato.

Com as informações do processo carregadas, são disponibilizadas opções para definição de iterações das atividades, cadastro de atores e associação destes com seus papéis, definição de prazo para as tarefas, indicação de ferramenta pra confecção dos artefatos e caminho para *templates* destes artefatos.

Para a definição de iterações, o mecanismo exibe a lista de atividades presentes no modelo e permite que o usuário selecione uma destas atividades e crie uma iteração para a mesma. Também é permitido que o usuário remova determinada atividade do processo. Na FIG 5.20 é exibida a tela de definição de iterações, onde foi criada uma iteração para a fase de construção.

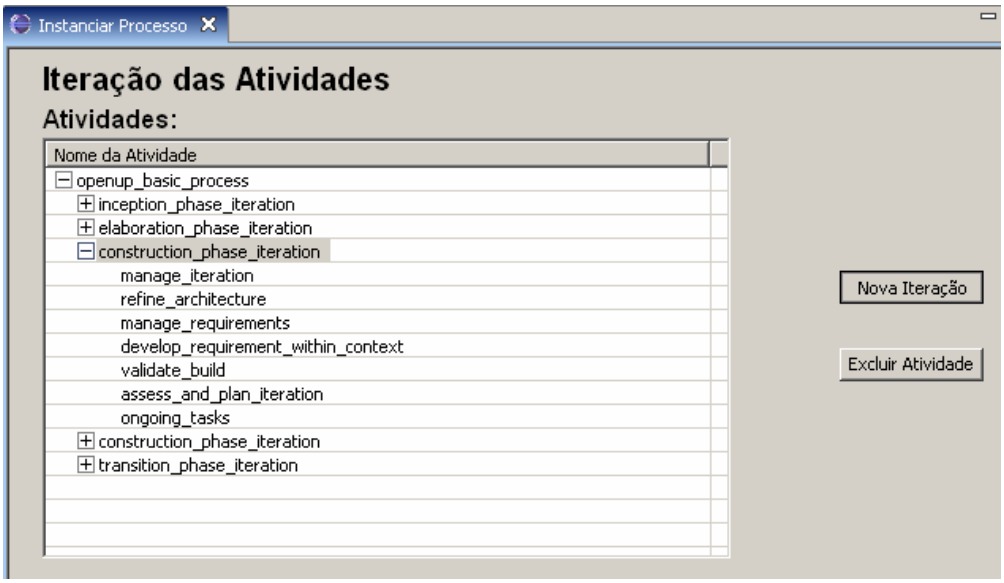


FIG 5.20 – Tela para definição de iterações.

Para as tarefas, deve-se definir a data de início e fim para cada uma delas. Elas se encontram agrupadas em suas respectivas atividades e podem ser vistas por meio de uma estrutura de árvore. A FIG 5.21 exibe um exemplo da tela para definição dos prazos deste exemplo, onde se observa a estrutura do processo e as datas de algumas tarefas.

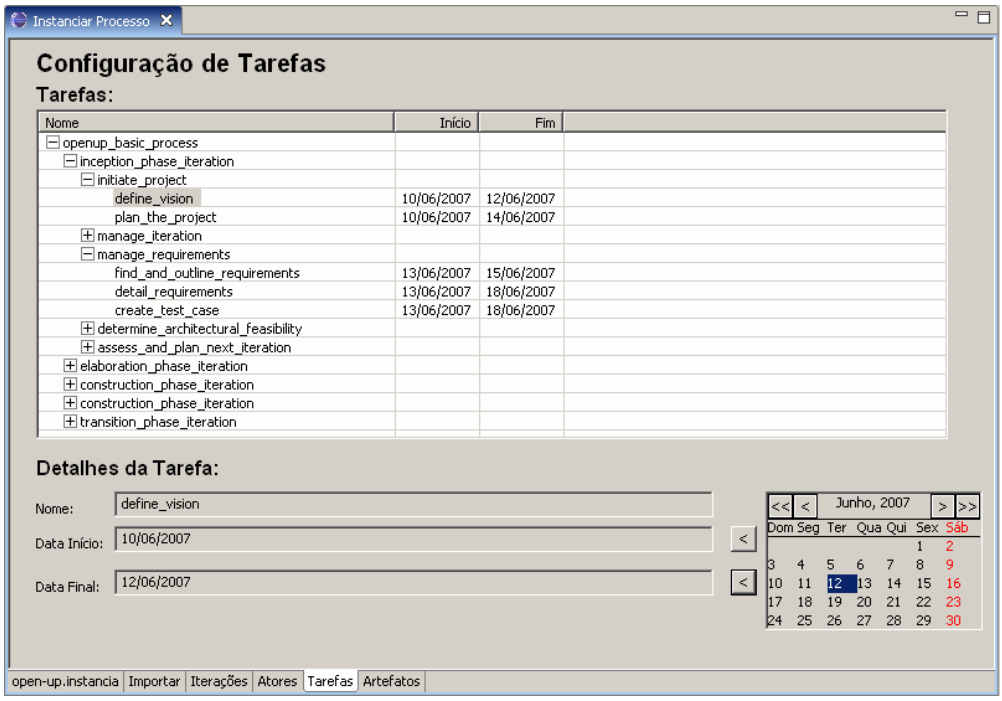


FIG 5.21 – Tela para definição de prazos das tarefas.

A definição de atores se dá com a indicação do nome e e-mail, além da relação de papéis ao qual este ator faz parte. Na FIG 5.22 é possível visualizar a definição de um ator responsável pelo desenvolvimento neste estudo de caso.

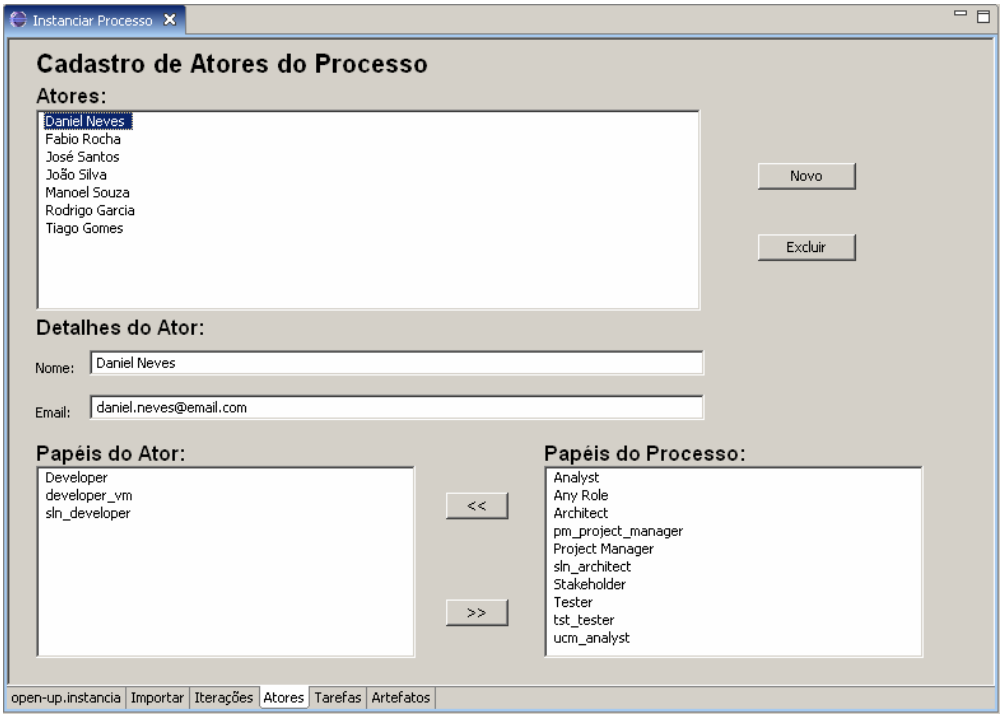


FIG 5.22 – Tela para definição de atores.

Os últimos elementos que faltam ser instanciados são os artefatos. Para cada um deles pode ser definido um conjunto de ferramentas que devem ser utilizadas para sua elaboração, e o caminho onde exista um *template* deste documento. Estas informações irão orientar o ator na geração deste artefato. Na FIG 5.23 é apresentada a tela que exibe a instanciação do Plano de Projeto neste estudo de caso.

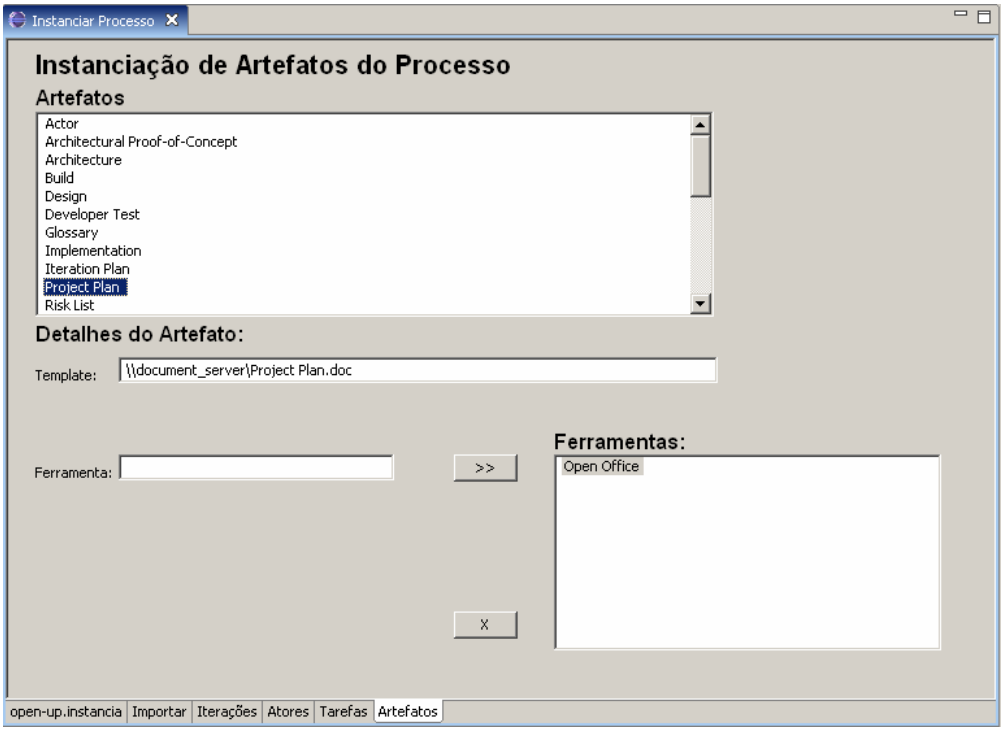


FIG 5.23 – Tela para definição de dados do artefato.

Com o cadastramento de todos estes dados, o mecanismo transforma o modelo abstrato em um modelo executável do processo. Este modelo executável é um arquivo XML que contém as informações do produto da fase de modelagem, mais as informações definidas nesta fase. Abaixo um pequeno trecho deste produto com os dados de instanciação de um ator, de uma tarefa e de um artefato.

```
<Instancia>
...
    <Actor name="Daniel Neves" email="daniel.neves@email.com" id="_igUi5tdNGNtdhZyYLXYuQX" >
      <Role>_0YDosMlgEdmt3adZL5Dmdw</Role>
      <Role>_sypCEPvjEdqf0-top1XJIg</Role>
      <Role>_BCtiMDR9EduwLdLujGQAIQ</Role>
    </Actor>
...
    <TaskInstance name="define_vision"
     id="_xupMvxOKEduCNqgZdt_OaA,_0oSdE8lgEdmt3adZL5Dmdw,_79bQ4DoCEdu0yYZ2bsCXog"
     idDescriptor="_79bQ4DoCEdu0yYZ2bsCXog" startDate="10/06/2007" finishDate="12/06/2007" >
        <Task>_0fOAoMlgEdmt3adZL5Dmdw</Task>
    </TaskInstance>
...
    <Artifact name="Project Plan" id="_0a6vcMlgEdmt3adZL5Dmdw"
     templatePath="\\document_server\Project Plan.doc">
        <Tool>Open Office</Tool>
    </Artifact>
...
</Instancia>
```

5.1.3. EXECUÇÃO

A fase de execução utiliza os dados do modelo executável gerado pela fase de instanciação para coordenar o andamento de um projeto e guiar os atores ao longo de seu andamento. As informações são carregadas e disponibilizadas para os atores com o objetivo de fornecer os dados necessários para a realização de cada uma de suas tarefas.

O mecanismo de execução é um sistema Web que permite a execução de diversos processos simultaneamente e possui dois perfis de acesso: gerente e ator. Desenvolvido utilizando tecnologia Java (Servlets e JSP) e banco de dados MySQL, o mecanismo funciona como um portal onde o usuário necessita de uma conta e senha para acessar o sistema.

Para iniciar um processo, o gerente do projeto deve, inicialmente, se cadastrar no sistema e criar sua conta e senha de acesso. Ao se cadastrar, o usuário passa automaticamente a assumir o papel de gerente e com isso são disponibilizados os recursos relacionados a este perfil.

O gerente deve utilizar o produto gerado pelo mecanismo de instanciação para fornecer ao mecanismo de execução os dados necessários para que se inicie a execução. Isto ocorre com a importação do arquivo que contém o modelo instanciado (FIG 5.24) no mecanismo de execução, que irá carregar os dados do processo e de sua instância.

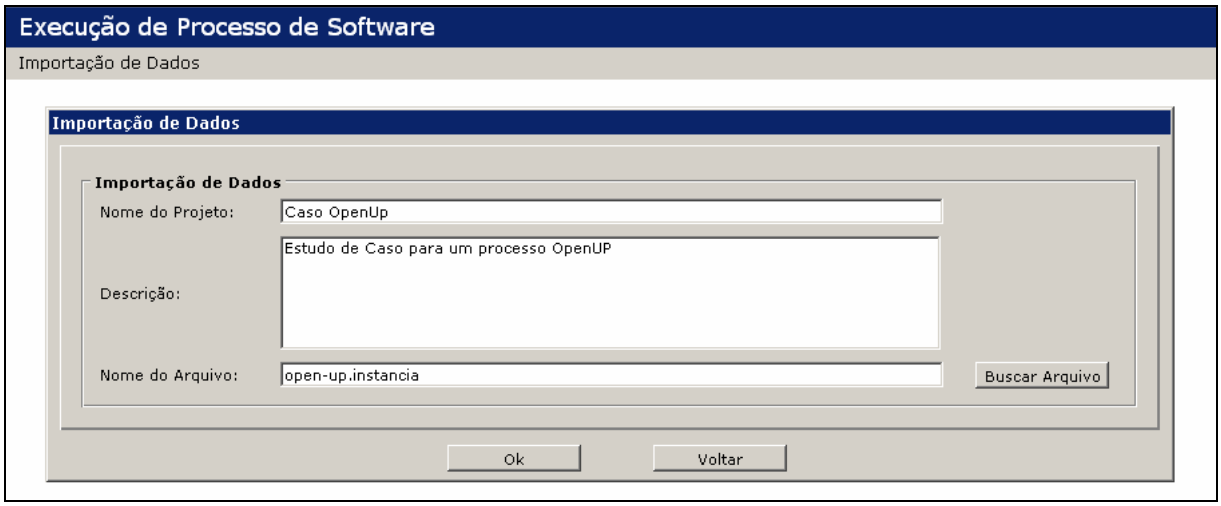


FIG 5.24 – Tela para importação do modelo instanciado.

Neste momento a execução do processo se inicia e o mecanismo de execução cria uma conta de acesso e uma senha para cada um dos sete atores definidos na instância do processo, enviando um e-mail para eles informando seu cadastro e os dados necessários para acessarem o sistema.

Ao acessar o sistema, o mecanismo irá verificar o processo relacionado a este ator e os papéis que o mesmo desempenha neste processo. Com isso, cada ator terá a possibilidade de visualizar a sua agenda de tarefas e a lista de tarefas completa do projeto. Esta lista é apresentada como um Gráfico de Gantt que ainda exibe o nome, os executores, porcentagem de conclusão e as datas de início e fim para cada tarefa. Uma parte da agenda do ator Daniel Neves é apresentada na FIG 5.25, onde podemos observar que a tarefa "*Manage Iteration*" será apresentada em vermelho indicando que se encontra atrasada, a tarefa "*Demonstrate the Architecture*" será apresentada na cor verde indicando que está concluída, a tarefa "*Plan Iteration*" será apresentada em azul indicando que a mesma se encontra em execução e no prazo, o restante será apresentada na cor cinza indicando que se encontram no estado de não iniciada.

FIG 5.25 – Trecho da agenda do ator.

O mecanismo possui um processo interno que é executado diariamente e verifica entre todos os atores de todos os processos, quais as tarefas atrasadas e as que deve ser iniciadas ou finalizadas no dia em questão, notificando os atores responsáveis pelas tarefas a informação correspondente. Como o ator possui uma tarefa com o estado “atrasada”, ele recebe uma notificação por e-mail informando que a tarefa já deveria ter sido encerrada. Esta notificação lhe é útil por lembrar que, apesar de ter encerrado a realização da tarefa, havia esquecido de registrar esta informação no sistema.

Cada tarefa da lista é um link que direciona o ator para visualizar a tela de detalhes da tarefa selecionada. Nesta tela, além dos dados da tarefa definidos na modelagem e instanciação, são exibidos os passos a serem seguidos para realização da mesma (FIG 5.26).

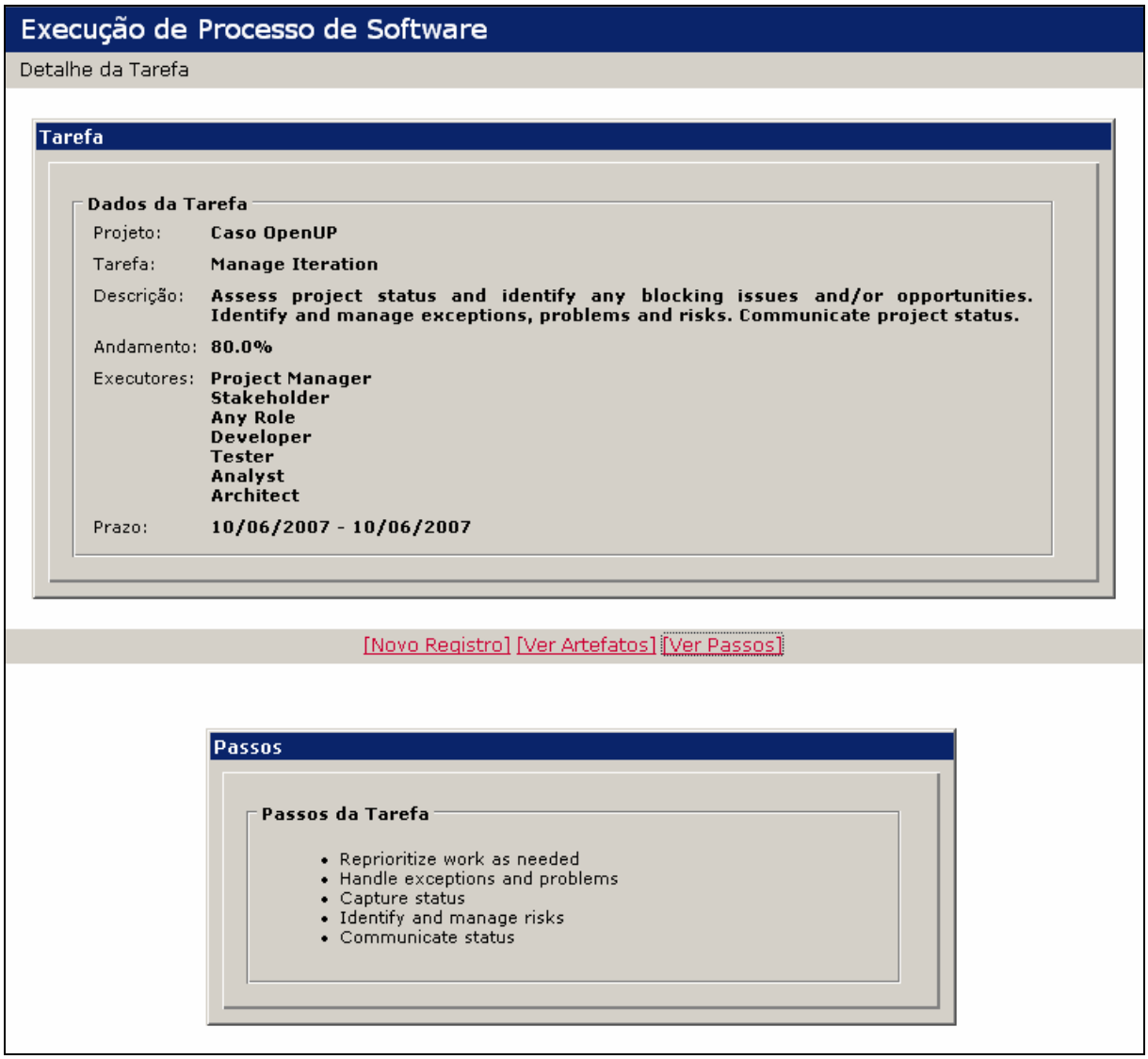


FIG 5.26 – Detalhes da tarefa.

Nesta mesma tela de detalhes, o ator tem a possibilidade de navegar em algumas informações da própria tarefa utilizando as opções localizadas no centro da tela. Uma destas opções é a visualização dos artefatos relacionados à tarefa em questão, que são separados como artefatos de entrada (obrigatório e opcional) e artefatos de saída (FIG 5.27).

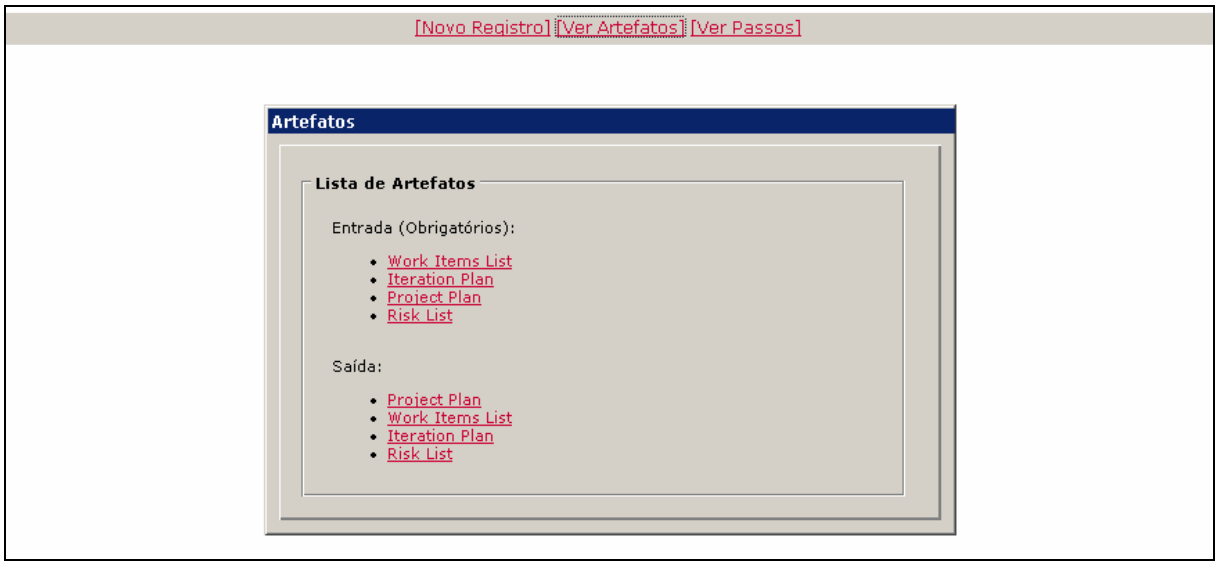


FIG 5.27 – Lista de artefatos da tarefa.

Na lista dos artefatos associados à tarefa, o ator tem a opção de acessar a sua tela de detalhes, o que possibilita não só visualizar os dados relativos a este artefato, mas também permite o acesso às versões deste documento e à inclusão de novas versões. Um exemplo da tela de detalhes do artefato "*Project Plan*" é apresentado na FIG 5.28

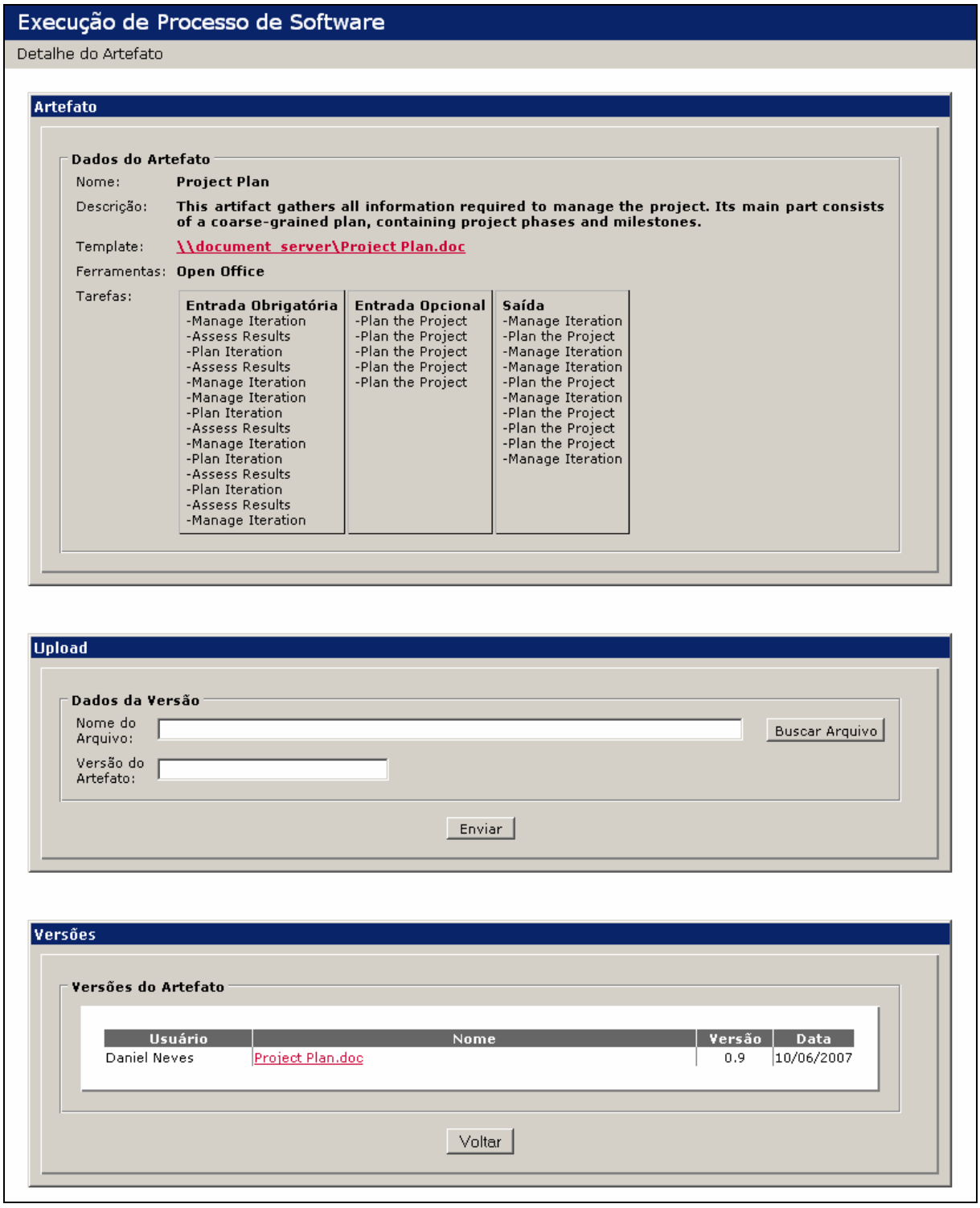


FIG 5.28 – Tela de detalhes do artefato *Project Plan*.

Outra opção da tela de detalhes da tarefa é a de criação de novos registros de ocorrências. Esta opção permite a visualização dos registros existentes e criação de uma nova ocorrência onde se deve registrar a data, o tempo utilizado, a porcentagem de conclusão e uma descrição do trabalho realizado (FIG 5.29).

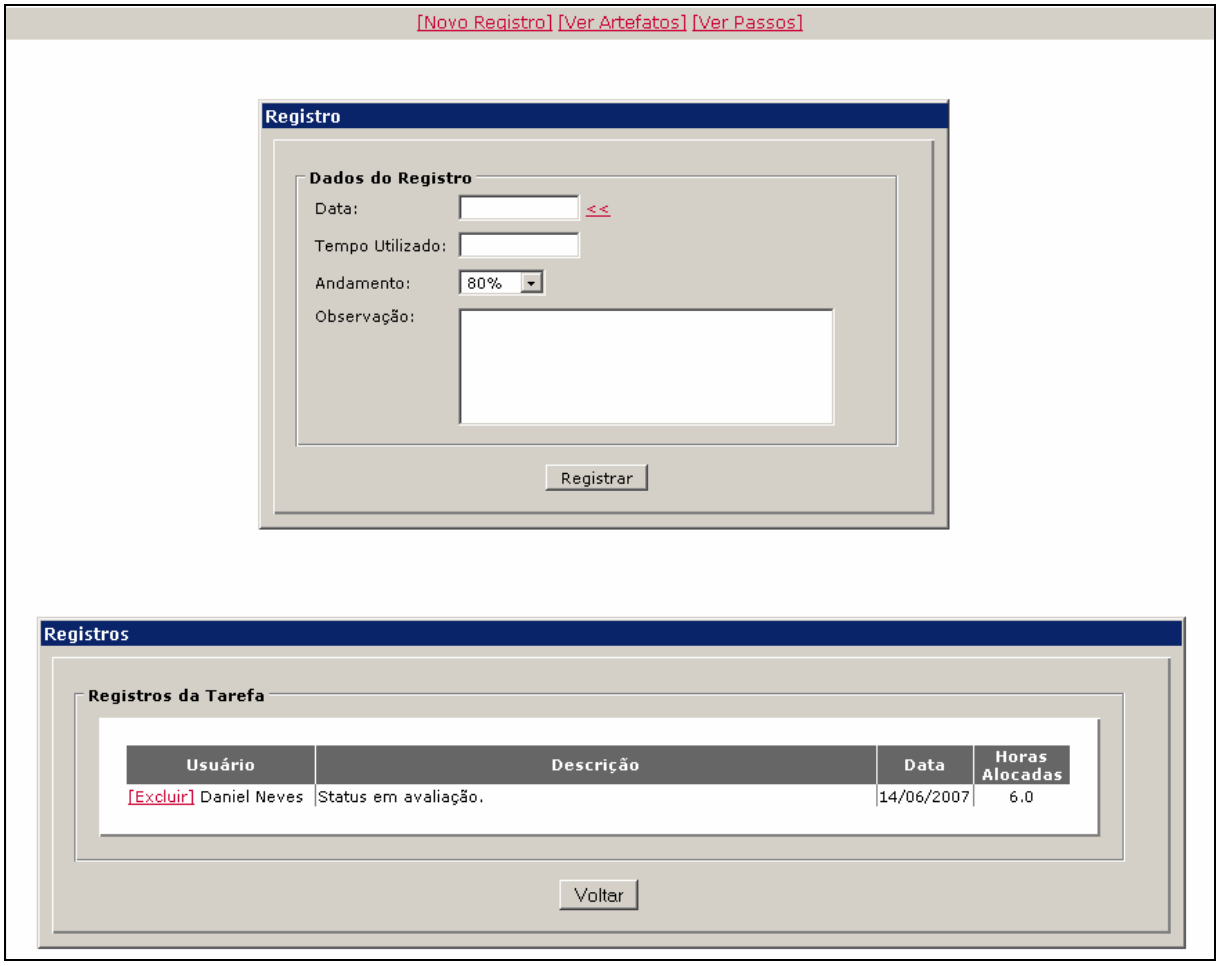


FIG 5.29 – Registro de ocorrências.

Este registro de ocorrências será importante para o gerente do processo, que tem a opção de consultar os registros de ocorrência de um determinado ator em um determinado período de tempo. Esta informação auxilia o gerente a reavaliar estimativas de prazo, que pode culminar em uma realocação da tarefa ou redefinição do prazo. A tela de relatório de horas é exibida na FIG 5.30.

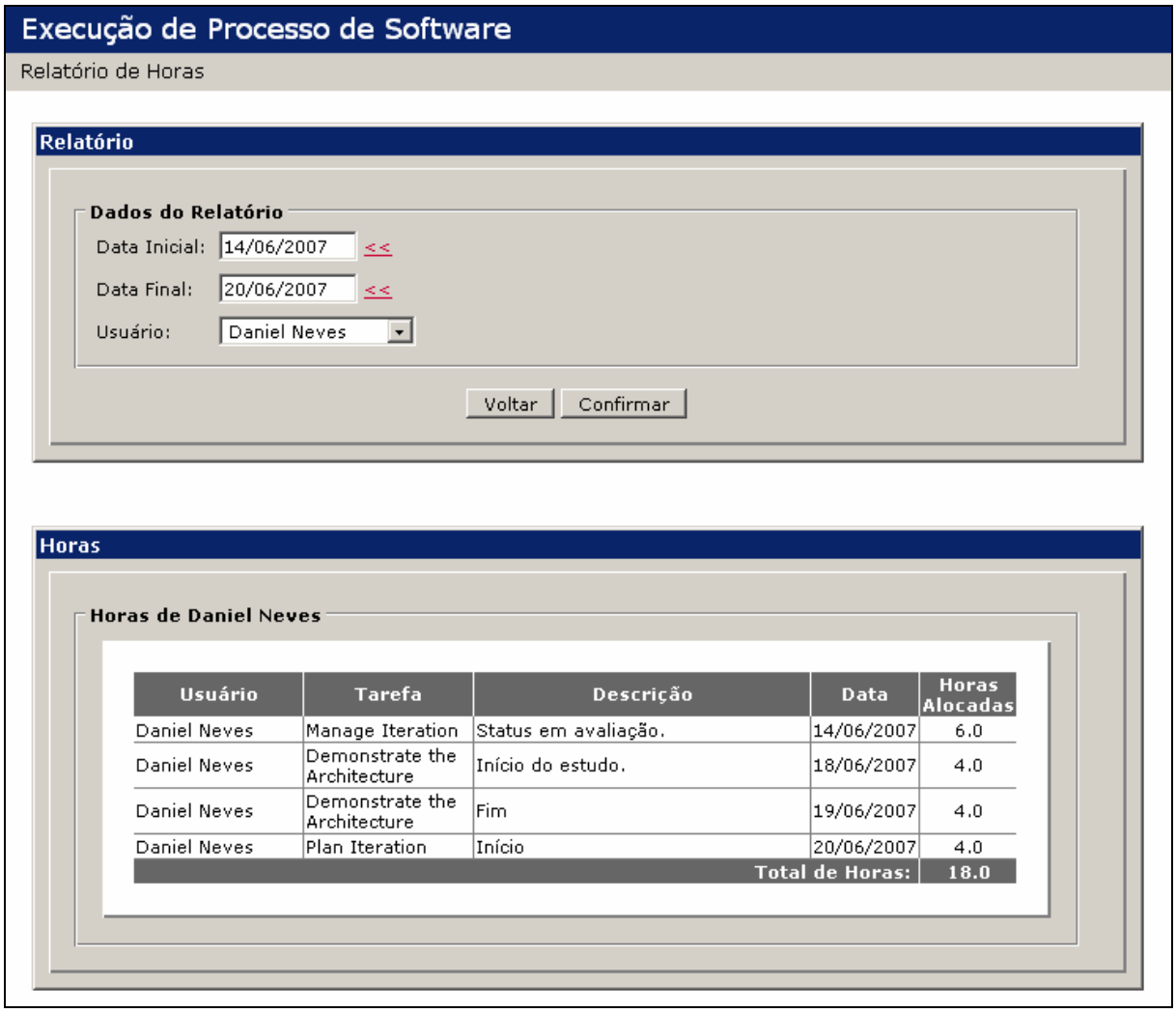


FIG 5.30 – Tela de relatório de horas.

5.1.4. LIÇÕES APRENDIDAS

Após a realização desta prova de conceito foi possível avaliar bem as transições entre as fases e a forma como o ambiente se comporta em algumas situações. Porém, devido ao fato do *OpenUP* ser um processo com diversas atividades e papéis, o controle por parte dos atores e dos prazos ficou ligeiramente comprometido. Por este motivo chegou-se a conclusão que seria melhor realizar uma nova prova de conceito com um outro modelo de processo, desta vez aplicando os conceitos aprendidos no *OpenUP* para modelar este novo processo.

5.2. MÉTODO DE DESENVOLVIMENTO DE SISTEMAS

Este estudo de caso irá utilizar o Método de Desenvolvimento de Sistemas (MDS), criado pela Diretoria de Sistemas de Informação do PRODERJ com o objetivo de ser adotado pelas

áreas desta autarquia, que trabalham desenvolvendo projetos que utilizam tecnologia da informação. O processo está estruturado em sete fases: Plano Sumário, Definição de Requisitos, Ante Projeto, Definição do Sistema por Módulo, Construção do Sistema, Implantação e Operação, e cada fase com suas correspondentes tarefas.

Neste trabalho, a última fase (Operação) não será considerada por não estar documentada na versão da especificação do processo que foi disponibilizada (Versão 3.0). Para uma melhor visibilidade na ferramenta de modelagem, optou-se por considerar estas seis fases como sub-processos que irão compor outros quatro processos: Análise, Projeto, Construção e Implantação.

5.2.1. MODELAGEM

O documento de especificação está estruturado de tal forma que, cada tarefa integrante das respectivas fases do MDS está estruturada com sua descrição; com os requisitos para a sua execução (artefatos de entrada); com os eventos da própria tarefa (passos); com os produtos da tarefa (artefato de saída); seguidos do suporte para a sua execução, como ferramentas computacionais e o responsável pela execução da tarefa (papel).

As fases que serão modeladas neste estudo de caso são:

- **Plano Sumário**: Servir como linha de base para o trabalho do gerente do projeto. É o documento que reconhece a existência de um projeto.
- **Definição de Requisitos**: A partir de uma solicitação efetuada externa ou internamente definir os requisitos necessários para uma proposta viável de desenvolvimento de um sistema que utiliza tecnologia da informação. Isto é feito através da análise do negócio do cliente e da verificação do sistema atual, levando a confecção de um modelo do sistema proposto.
- **Ante Projeto**: Definir as características lógicas do sistema proposto, como a delimitação de seu escopo, especificação da arquitetura, modelo de classes (modelo conceitual dos dados e modelo de funções) e de interfaces com sistemas existentes, definindo ainda a equipe multidisciplinar que atuará no projeto, a infra-estrutura operacional e as estimativas para o desenvolvimento do projeto.
- **Definição do Sistema**: Transformar as características lógicas anteriormente definidas em modelo físico de dados, modelo de estrutura funcional e

especificações técnicas das funções, definindo ainda as características operacionais, entradas, saídas e interfaces.

- **Construção do Sistema**: Após a geração das tabelas, com seus respectivos acessos, da implementação das interfaces com os sistemas legados e conclusão das especificações técnicas das funções, construir o sistema através da codificação dos módulos e rotinas, testando-os individualmente e de forma integrada, elaborando paralelamente os manuais que possibilitarão a operação futura do sistema e ao final, elaborar planejamento da implantação do sistema.
- **Implantação do Sistema**: Executar as etapas de treinamento do pessoal responsável pela operação do sistema, a preparação do ambiente de produção assegurando a disponibilidade de hardware e software, povoamento das bases de dados, condução de testes de aceitação e a transferência do sistema para os usuários e pessoal de apoio.

Estas seis fases foram agrupadas em outros quatro processos que resultaram na seguinte organização:

- **Análise**: Plano Sumário, Definição de Requisitos e Ante Projeto (FIG 5.31);
- **Projeto**: Definição do Sistema (FIG 5.32);
- **Construção**: Construção do Sistema (FIG 5.33);
- **Implantação**: Implantação do Sistema (FIG 5.34).

FIG 5.31 – Tela com diagrama de atividades do processo de Análise.

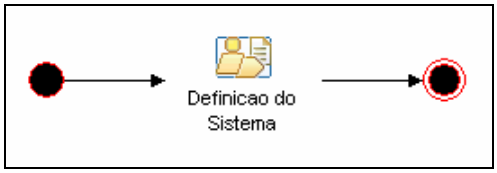


FIG 5.32 – Tela com diagrama de atividades do processo de Projeto.

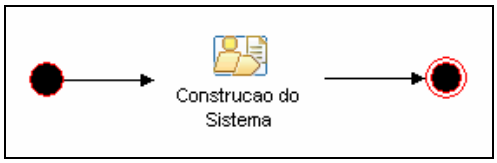


FIG 5.33 – Tela com diagrama de atividades do processo de Construção.

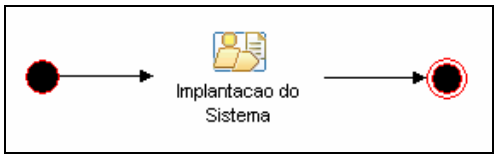


FIG 5.34 – Tela com diagrama de atividades do processo de Implantação.

Para cada fase foi criado um pacote na pasta *Method Content* e em cada pacote foram modeladas suas respectivas tarefas. Os papéis e os artefatos foram modelados no pacote correspondente a fase onde há a sua primeira ocorrência no processo. Um exemplo da organização dos pacotes é apresentado na FIG 5.35.

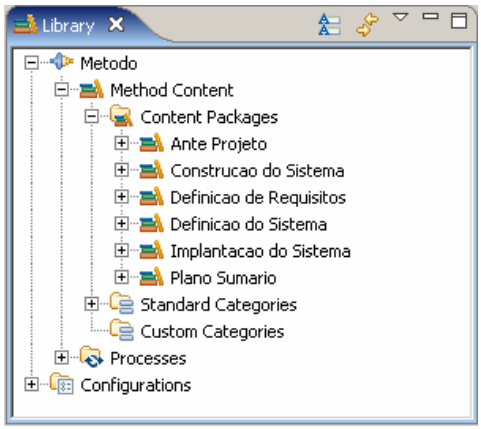


FIG 5.35 – Pacotes do conteúdo reutilizável.

Para explicitar a forma como o documento está estruturado, e como esta especificação se reflete na ferramenta de modelagem, abaixo é apresentado a especificação de uma tarefa e, em seguida, as telas desta tarefa modelada no EPF. Na FIG 5.36 é apresentada a tela de descrição, na FIG 5.37 a tela com os passos, na FIG 5.38 a tela com os executores e na FIG 5.39 a tela com os artefatos.

Entendimento do Modelo do Negócio

Descrição: Nesta fase serão conhecidos os requisitos necessários à elaboração do projeto a ser implementado através de entrevistas com o usuário. Nestas entrevistas iremos adquirir o conhecimento necessário para identificar as principais funções do negócio do usuário, seus problemas e as necessidades.

Requisitos: Solicitação do Serviço.

Eventos:

Definir as metas do sistema;
Especificar as necessidades cruciais para o sucesso;
Citar principais funções do negócio;
Definir as áreas do usuário envolvidas no projeto;
Diferenciar as necessidades dos desejos do usuário;
Definir envolvimento de cada área;
Relatar o desempenho desejado;
Especificar as questões de confiabilidade;
Relacionar os requisitos de produção;
Especificar a tecnologia necessária;
Construir organograma;
Citar os sistemas / subsistemas antigos envolvidos;
Relatar as extensões futuras.

Produtos: Modelo do Negócio.

Suporte: Editor de Texto.

Responsável: Analista de Negócios.

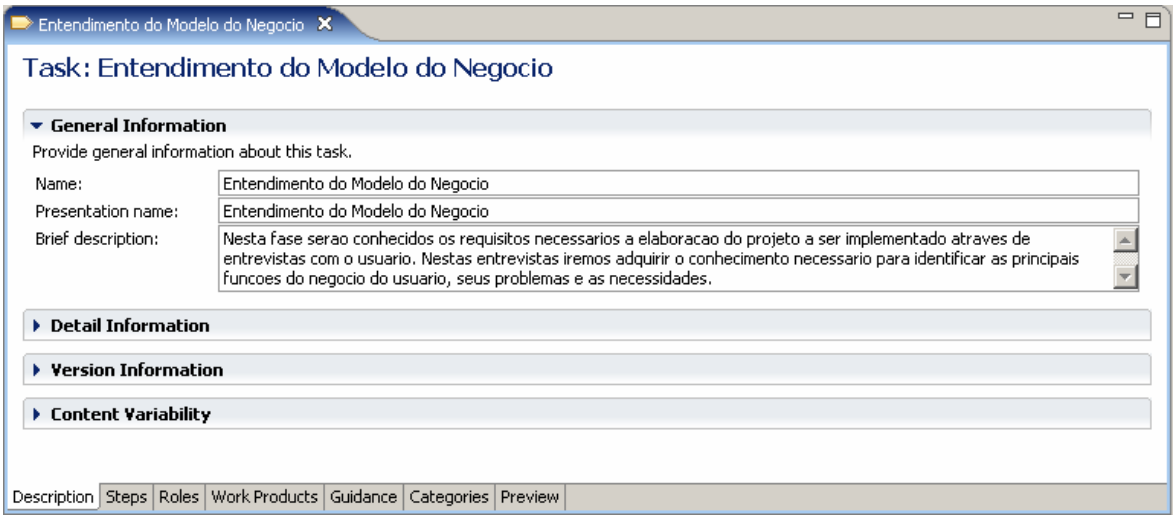


FIG 5.36 – Tela de descrição da tarefa.

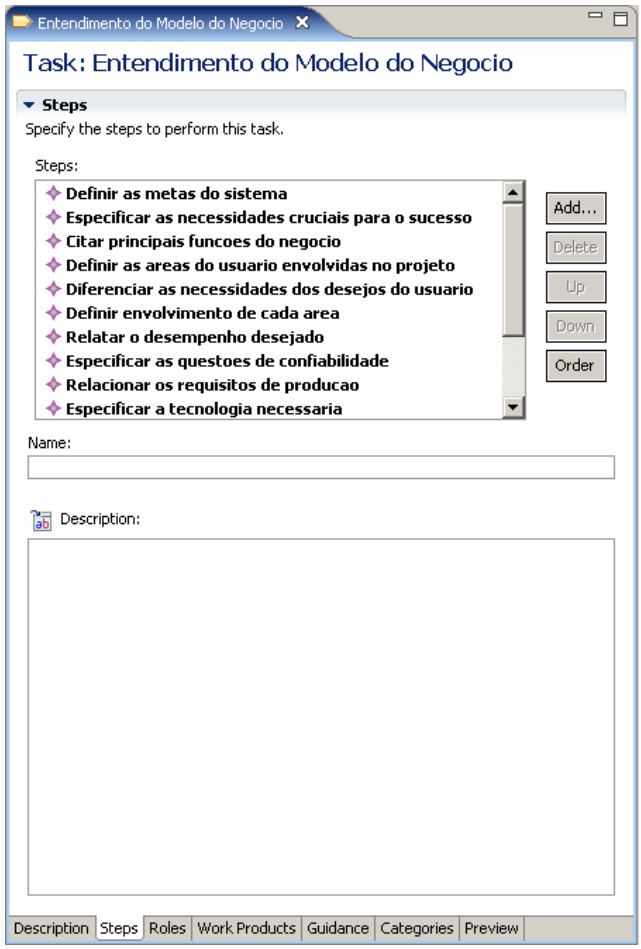


FIG 5.37 – Tela com passos da tarefa.

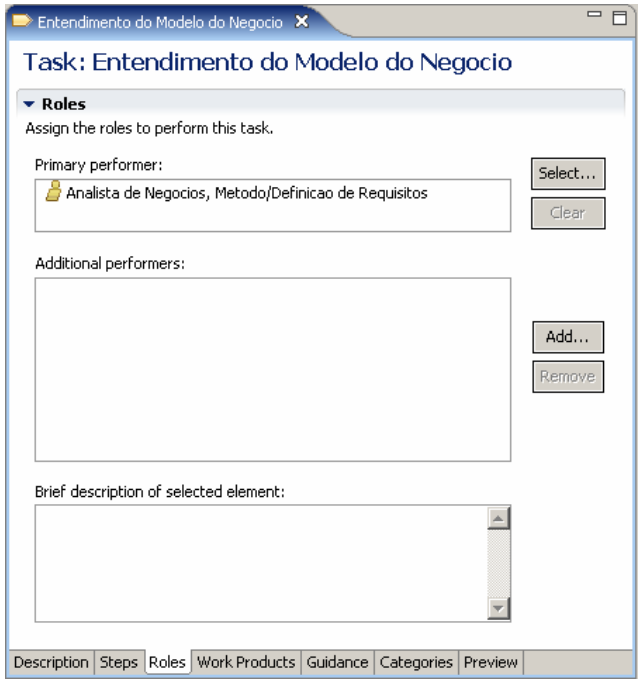


FIG 5.38 – Tela com executores da tarefa.

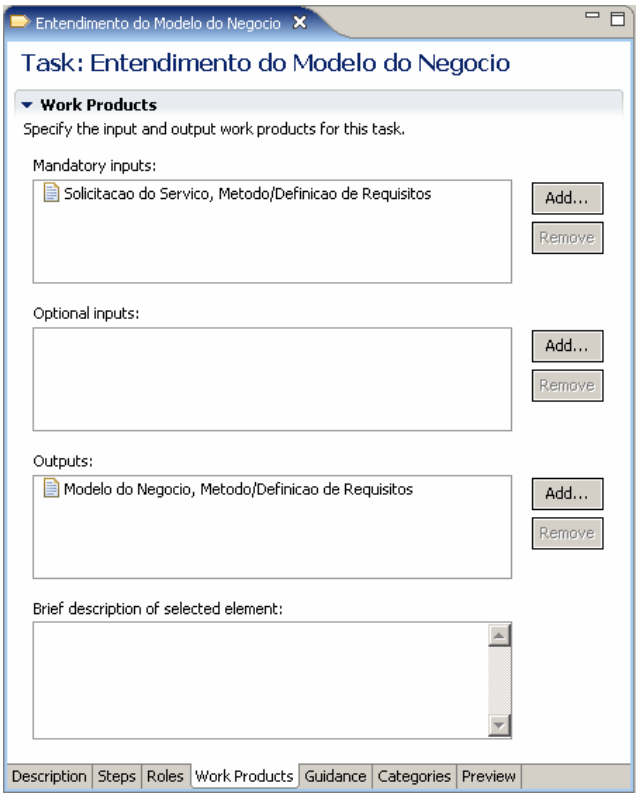


FIG 5.39 – Tela com artefatos da tarefa.

Na pasta *Capability Patterns*, foram criadas duas pastas: uma para comportar as fases do processo e outra para os quatro agrupamentos utilizados para estas fases. Esta estrutura de pastas é exibida na FIG 5.40.

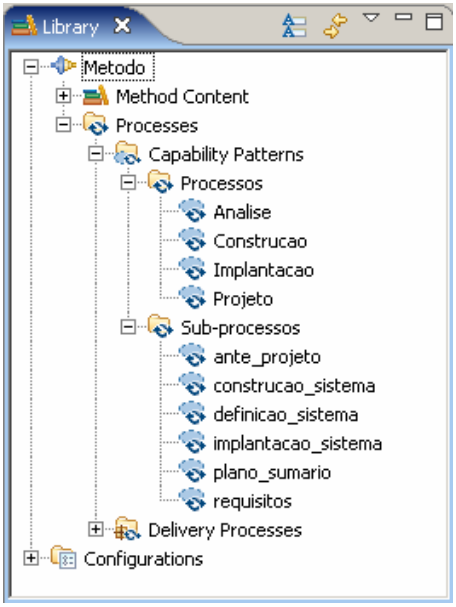
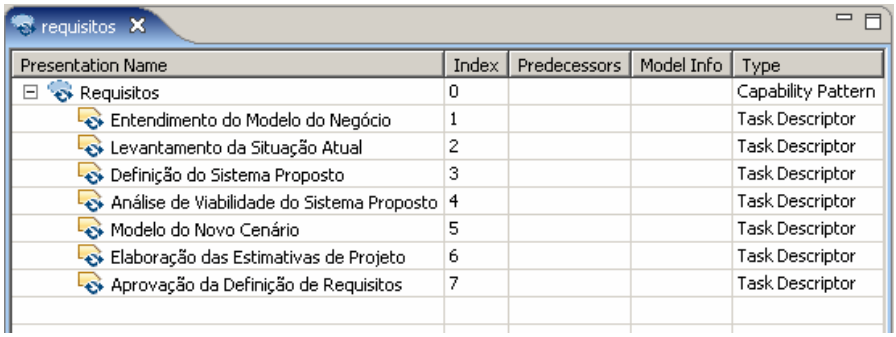


FIG 5.40 – Tela com lista de sub-processos e processos.

Para exemplificar a definição de uma fase do MDS, abaixo é apresentado o WBS da fase de definição de requisitos na FIG 5.41 e o seu respectivo diagrama de atividades na FIG 5.42.



Presentation Name	Index	Predecessors	Model Info	Type
Requisitos	0			Capability Pattern
Entendimento do Modelo do Negócio	1			Task Descriptor
Levantamento da Situação Atual	2			Task Descriptor
Definição do Sistema Proposto	3			Task Descriptor
Análise de Viabilidade do Sistema Proposto	4			Task Descriptor
Modelo do Novo Cenário	5			Task Descriptor
Elaboração das Estimativas de Projeto	6			Task Descriptor
Aprovação da Definição de Requisitos	7			Task Descriptor

FIG 5.41 – Tela com WBS da definição de requisitos.

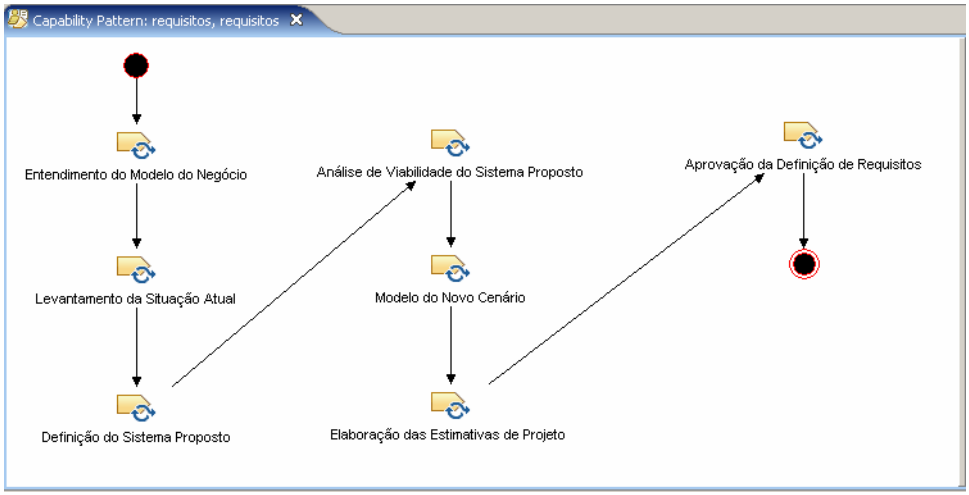
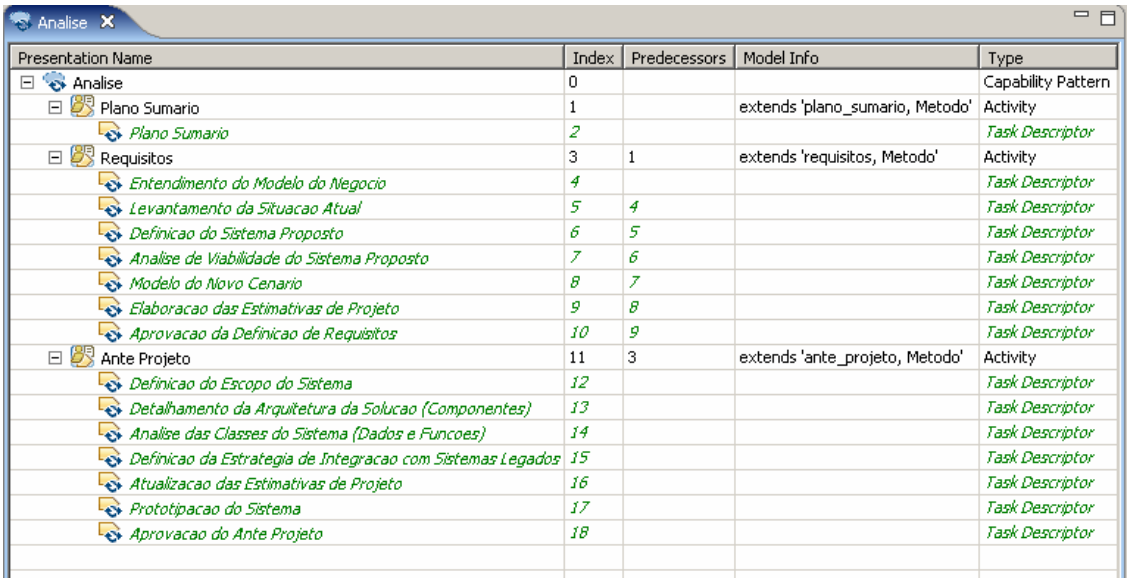


FIG 5.42 – Tela com diagrama de atividades da definição de requisitos.

Para exemplificar o agrupamento das fases do MDS nos quatro processos sugeridos neste estudo de caso, pode-se observar o WBS do processo de Análise na FIG 4.43.



Analise

Presentation Name	Index	Predecessors	Model Info	Type
Analise	0			Capability Pattern
Plano Sumario	1		extends 'plano_sumario, Metodo'	Activity
Plano Sumario	*2*			*Task Descriptor*
Requisitos	3	1	extends 'requisitos, Metodo'	Activity
Entendimento do Modelo do Negocio	*4*			*Task Descriptor*
Levantamento da Situacao Atual	*5*	*4*		*Task Descriptor*
Definicao do Sistema Proposto	*6*	*5*		*Task Descriptor*
Analise de Viabilidade do Sistema Proposto	*7*	*6*		*Task Descriptor*
Modelo do Novo Cenario	*8*	*7*		*Task Descriptor*
Elaboracao das Estimativas de Projeto	*9*	*8*		*Task Descriptor*
Aprovacao da Definicao de Requisitos	*10*	*9*		*Task Descriptor*
Ante Projeto	11	3	extends 'ante_projeto, Metodo'	Activity
Definicao do Escopo do Sistema	*12*			*Task Descriptor*
Detalhamento da Arquitetura da Solucao (Componentes)	*13*			*Task Descriptor*
Analise das Classes do Sistema (Dados e Funcoes)	*14*			*Task Descriptor*
Definicao da Estrategia de Integracao com Sistemas Legados	*15*			*Task Descriptor*
Atualizacao das Estimativas de Projeto	*16*			*Task Descriptor*
Prototipacao do Sistema	*17*			*Task Descriptor*
Aprovacao do Ante Projeto	*18*			*Task Descriptor*

FIG 5.43 – Tela com o WBS do processo de Análise.

Ao modelar as tarefas, foram identificados os papéis existentes no processo, que foram modelados em seus respectivos pacotes do *Method Content*. Os papéis do MDS são:

- Analista de Banco de Dados;
- Analista de Negócios;
- Analista de O&M;
- Analista de Sistemas;
- Analista de Sistemas da Gerência de Análise de Negócios;
- Analista de Sistemas das Gerências de Suporte;
- Coordenador do Projeto;
- Diretores do PRODERJ;
- Documentador de Sistemas;
- Gerentes das Áreas Envolvidas;
- Programador de Sistemas;
- Programador Visual;
- Projetista de Sistemas;
- Usuário Responsável;
- Usuário Solicitante;
- Usuário do Sistema.

Ainda modelando as tarefas, ao se identificar seus artefatos de entrada e saída, estes foram modelados no conteúdo reutilizável do processo. A lista de artefatos do processo é apresentada na FIG 5.44.

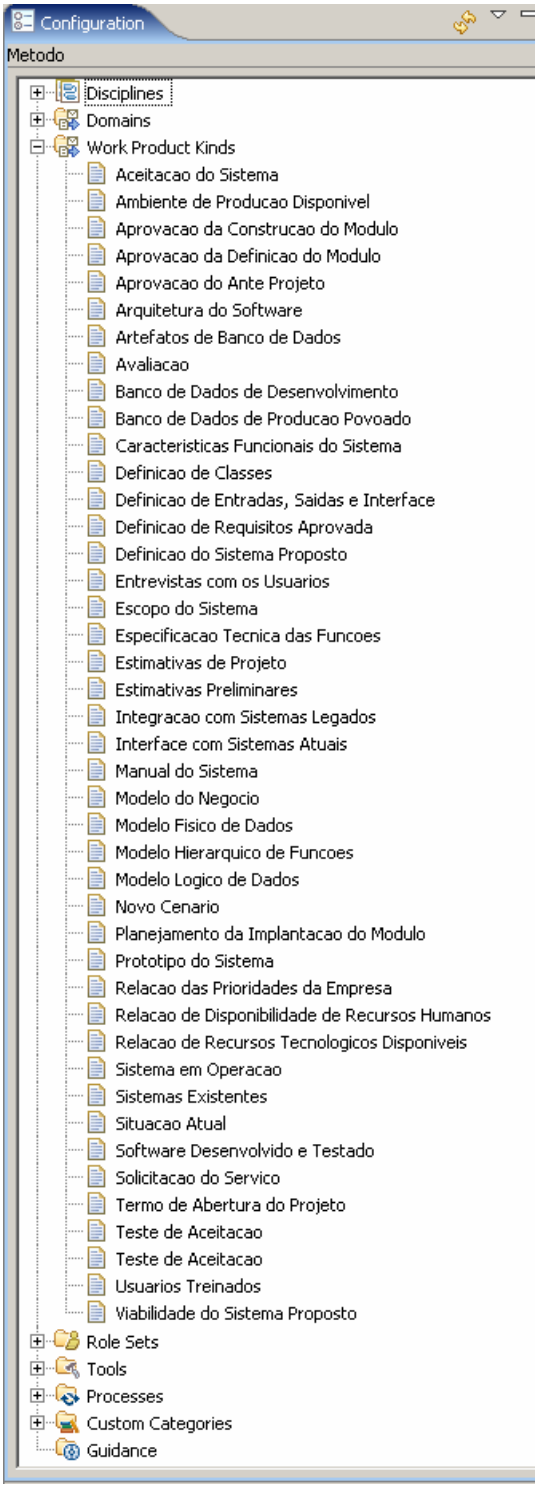


FIG 5.44 – Tela com lista de artefatos do MDS.

O processo do MDS será composto então pelos processos de Análise, Projeto, Construção e Implantação, que por sua vez são compostos pelas seis fases do Método. Na FIG 5.45 é exibido o WBS do Método de Desenvolvimento de Sistemas e na FIG 5.46 o seu diagrama de atividades.

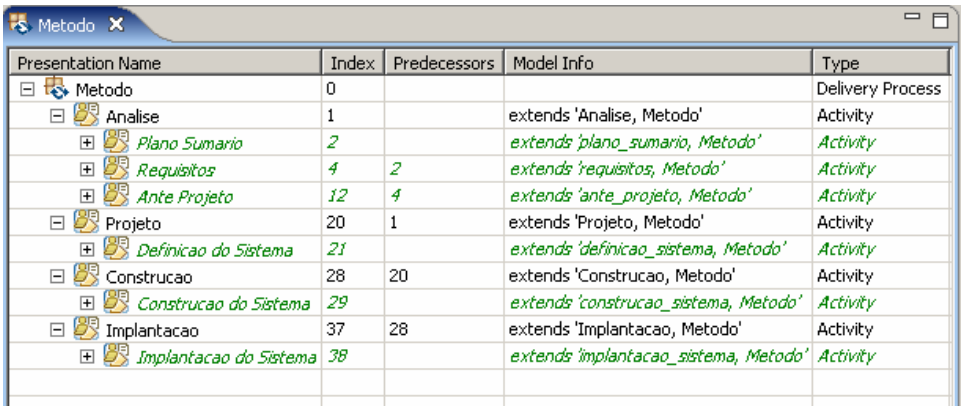


FIG 5.45 – WBS do Método.

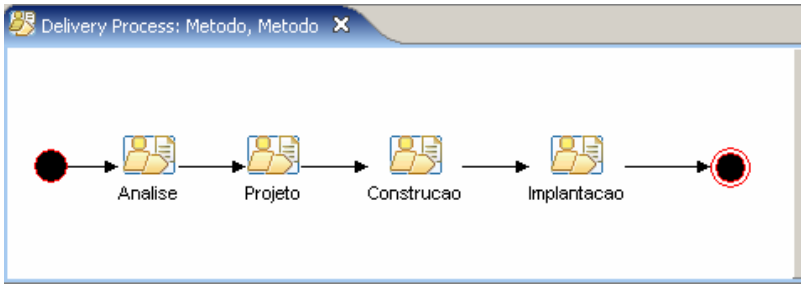


FIG 5.46 – Diagrama de atividades do Método.

Um trecho do produto gerado pelo EPF é apresentado abaixo:

```
<MethodPackage xsi:type="uma:ProcessComponent" name="Metodo" ... >
  <Process xsi:type="uma:DeliveryProcess" name="Metodo" ... >
  ...
    <BreakdownElement xsi:type="uma:Activity" name="Analise" ... >
      <SuperActivity>_smxU4VfbEduqLcQ6X4Xv-Q</SuperActivity>
    </BreakdownElement>
    <BreakdownElement xsi:type="uma:Activity" name="Projeto" ... >
      <SuperActivity>_smxU4VfbEduqLcQ6X4Xv-Q</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_3_QqgA1eEdy5y42d8npuEQ">
        _x7RUYVfbEduqLcQ6X4Xv-Q
      </Predecessor>
    </BreakdownElement>
    <BreakdownElement xsi:type="uma:Activity" name="Construcao" ... >
      <SuperActivity>_smxU4VfbEduqLcQ6X4Xv-Q</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_jPETYP2XEduo9q7snrRL2Q">
        _zWTvAVfbEduqLcQ6X4Xv-Q
      </Predecessor>
    </BreakdownElement>
    <BreakdownElement xsi:type="uma:Activity" name="Implantacao" ... >
      <SuperActivity>_smxU4VfbEduqLcQ6X4Xv-Q</SuperActivity>
      <Predecessor linkType="finishToFinish" id="_jblFYP2XEduo9q7snrRL2Q">
        _0lvWkVfbEduqLcQ6X4Xv-Q
      </Predecessor>
    </BreakdownElement>
  ...
  </Process>
</MethodPackage>
```

5.2.2. INSTANCIAÇÃO

A instanciação do processo se inicia com a importação do produto gerado pelo EPF no processo de instanciação criado. A tela de importação é exibida na FIG 5.47 onde se observa a importação do arquivo “Metodo.xml” no processo de instanciação “metodo.instancia”.

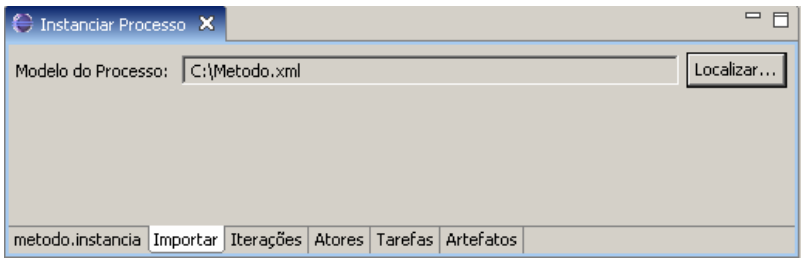


FIG 5.47 – Tela de importação do modelo abstrato.

Com fim da importação do produto, os dados presentes no modelo abstrato estão carregados e a instanciação do processo se inicia. Abrindo a aba “Tarefas” do editor, são exibidas as atividades e tarefas que compõem o processo e para cada tarefa são definidas data de início e fim. A FIG 5.48 exibe uma parte das tarefas instanciadas para este estudo de caso.

FIG 5.48 – Tela para instanciação de tarefas.

Abrindo a aba “Atores” do mecanismo, é exibida a tela com os papéis do processo, os atores instanciados e seus respectivos papéis. É permitido nesta tela inclusão e exclusão de atores, além da alteração do nome e e-mail do ator bem como a inclusão ou exclusão de papéis. Um exemplo da tela de instanciação de atores deste estudo de caso é apresentado na FIG 5.49.

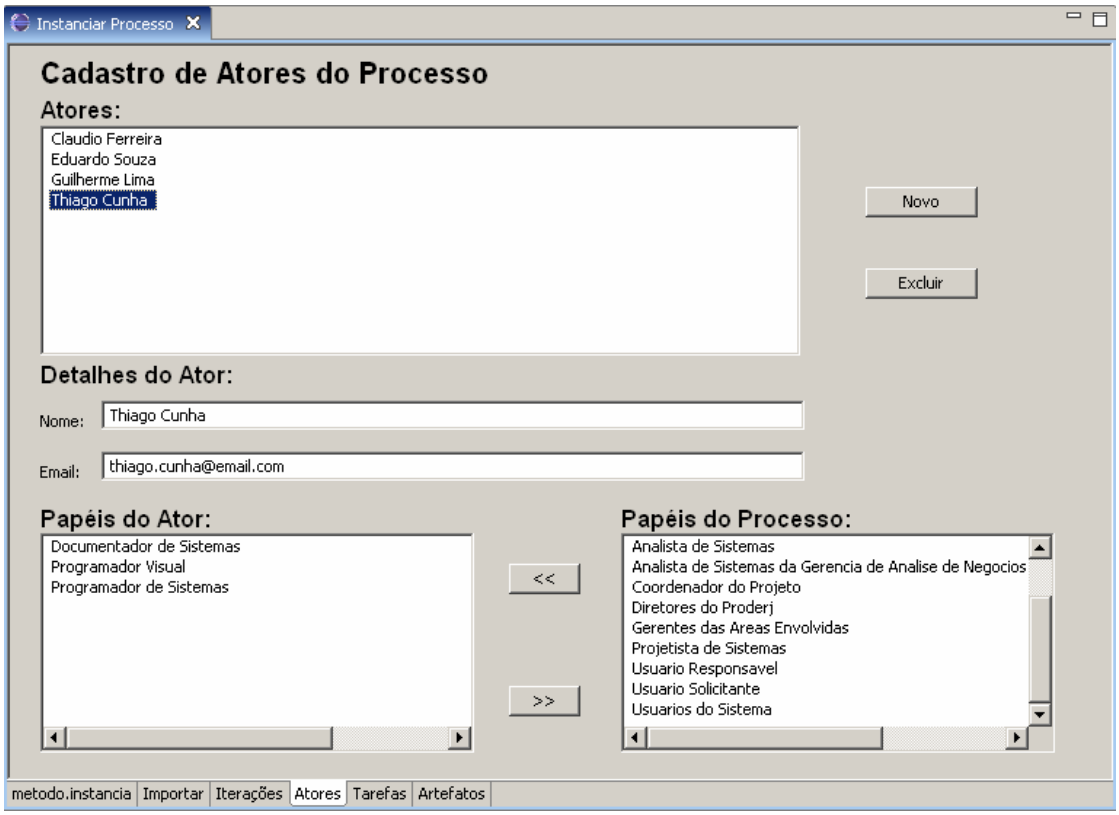


FIG 5.49 – Tela para instanciação de atores.

Os artefatos do processo são exibidos na aba “Artefatos” do mecanismo, e para cada um deles é permitida a definição de um caminho para o *template* e de uma lista de ferramentas que podem ser utilizadas na geração destes artefatos. A FIG 5.50 exemplifica a instanciação do artefato “Definição de Classes” do processo.

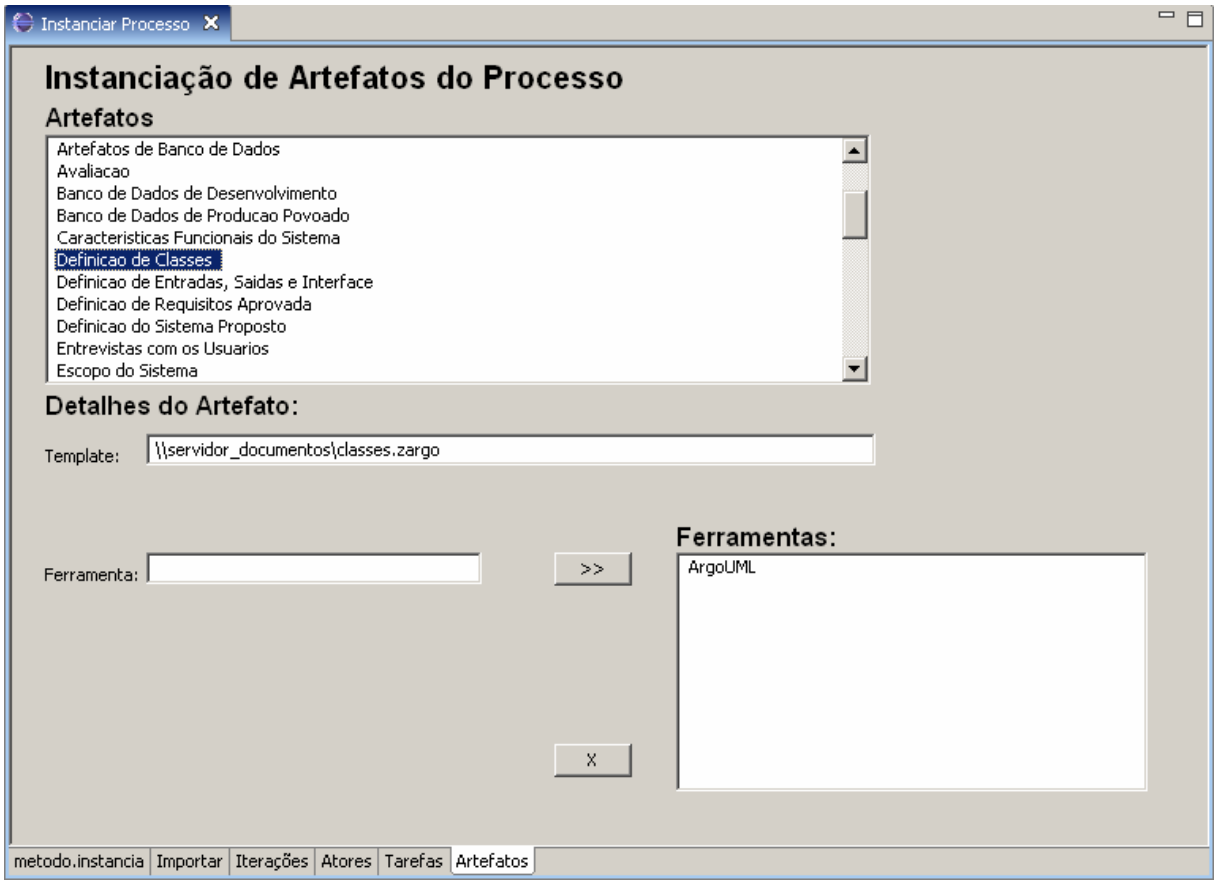


FIG 5.50 – Tela para instanciação de artefatos.

O arquivo “metodo.instancia” que é editado pelo mecanismo de instanciação é o próprio produto desta fase, e contém todos os dados do modelo abstrato e da instanciação. Apenas para exemplificação, é apresenta um trecho deste arquivo com a instanciação de alguns elementos.

```
<Instancia>
  ...
  <Actor name="Thiago Cunha" email="thiago.cunha@email.com"
   id="_Ypkwg8wluWF2OiicFV07QL" >
    <Role>_5OTisFDAEduvns6qUFPr_Q</Role>
    <Role>_15C6QFDEEduXSO2YyMBC3g</Role>
    <Role>_c2m7EFDFEduXSO2YyMBC3g</Role>
  </Actor>
  ...
  <TaskInstance name="Aprovacao da Definicao de Requisitos"
   id="_x7RUYVfbEduqLcQ6X4Xv-Q,_a1N-0VfbEduqLcQ6X4Xv-Q,_gQRSQVfbEduqLcQ6X4Xv-
   Q,_DX3whVC9Eduvns6qUFPr_Q" idDescriptor="_DX3whVC9Eduvns6qUFPr_Q"
   startDate="15/06/2007" finishDate="19/06/2007" >
    <Task>_5aPLoEnKEduTrOaMwD9MlQ</Task>
  </TaskInstance>
  ...
  <Artifact name="Definicao de Classes" id="_93u2EEtIEdufsPezJMXoOQ"
   templatePath="\\servidor_documentos\classes.zargo">
    <Tool>ArgoUML</Tool>
  </Artifact>
  ...
</Instancia>
```

5.2.3. EXECUÇÃO

Na tela de entrada do mecanismo de execução há a opção de entrada no sistema para usuários já cadastrados e a opção para que o usuário se cadastre (FIG 5.51). Para iniciar a execução do processo, o gerente se cadastra no sistema definindo seu nome, e-mail, conta e senha de acesso (FIG 5.52).

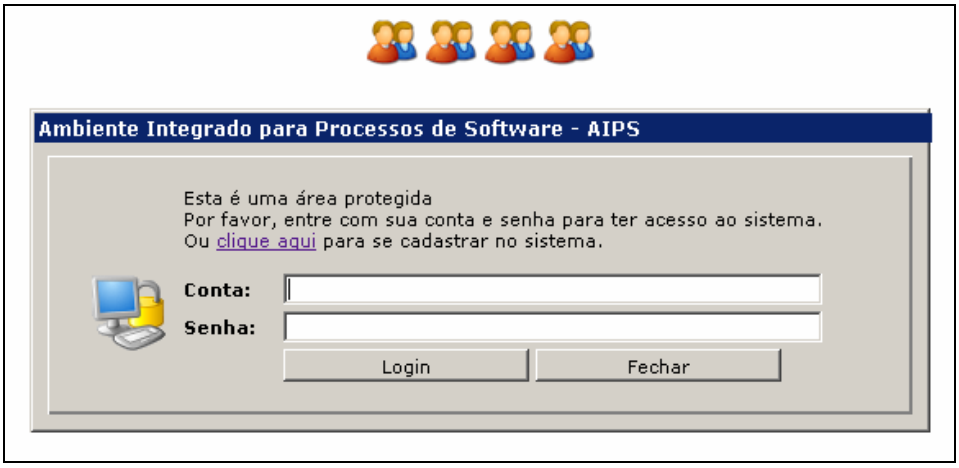


FIG 5.51 – Tela de entrada.

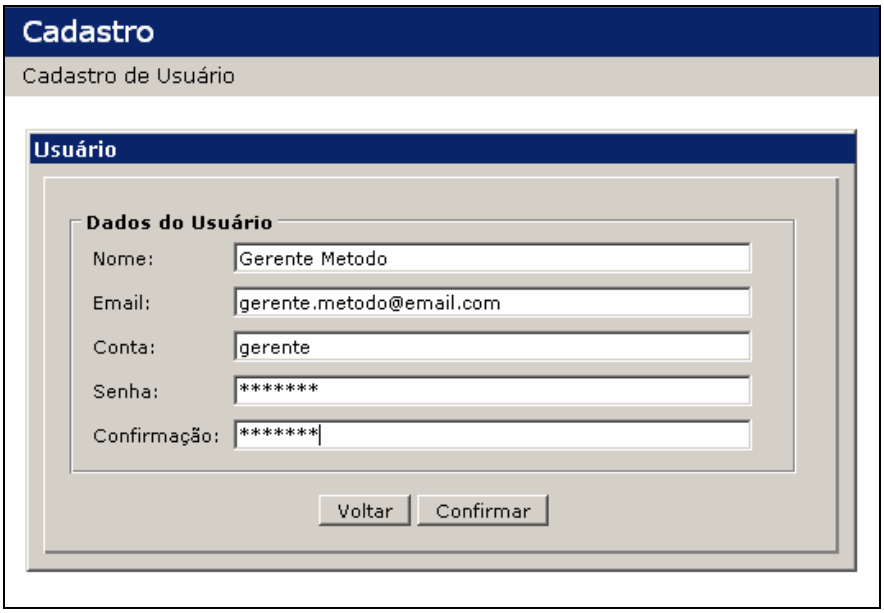


FIG 5.52 – Tela para cadastro de usuário.

Ao se cadastrar, o gerente entra no sistema e é levado à lista de opções para o seu perfil, que permite a importação dos dados, a alteração de senha e a saída do sistema. A FIG 5.53 apresenta a tela com a lista de opções para o gerente.

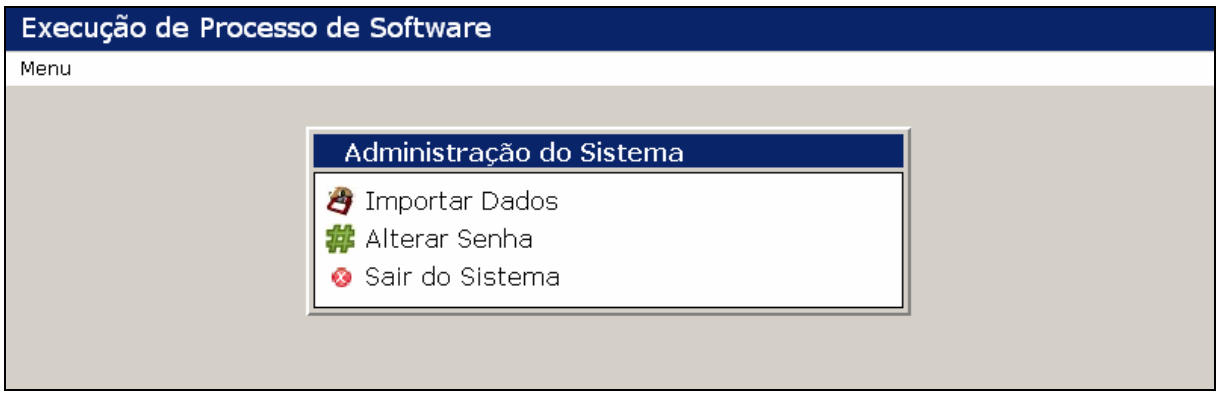


FIG 5.53 – Tela de opções do gerente.

Selecionando a opção "Importar Dados", o gerente é direcionado para a tela onde ele deve definir um nome e uma descrição para o projeto e enviar o produto gerado na instanciação para que os dados do processo contidos no arquivo sejam carregados e passem a ser controlados no sistema (FIG 5.54).

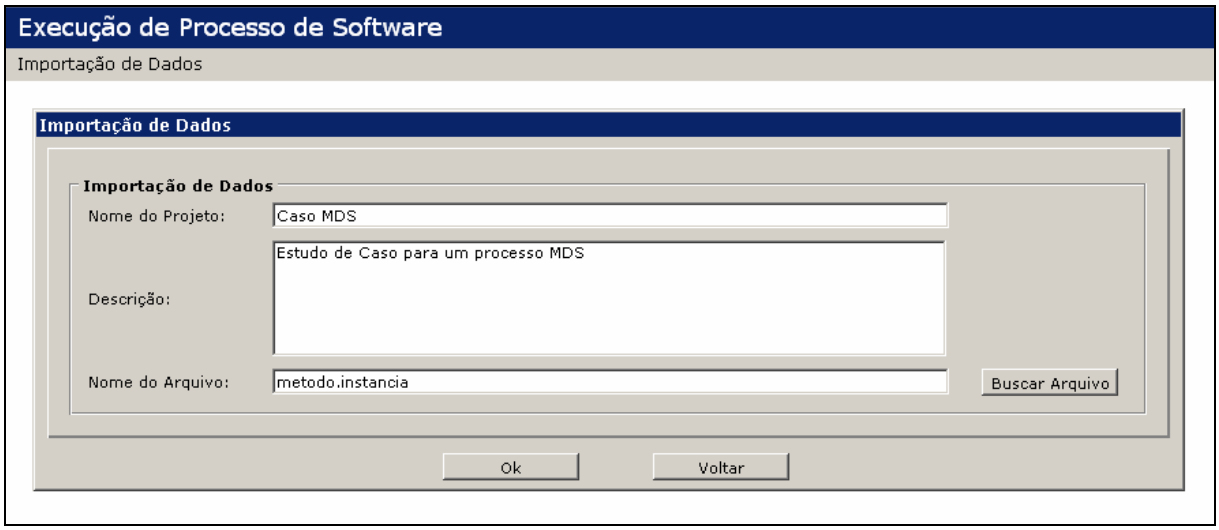


FIG 5.54 – Tela para importação de dados.

Ao ler as informações do modelo instanciado o sistema persiste todos os dados no banco de dados, cria contas de acesso para os atores e as envia por e-mail e inicia a execução do processo. Após o início da execução, o gerente não tem mais a opção de importação de dados

e duas novas opções são dadas: Relatório de Horas por Usuário e Exibir Gráfico do Projeto (FIG 5.55).

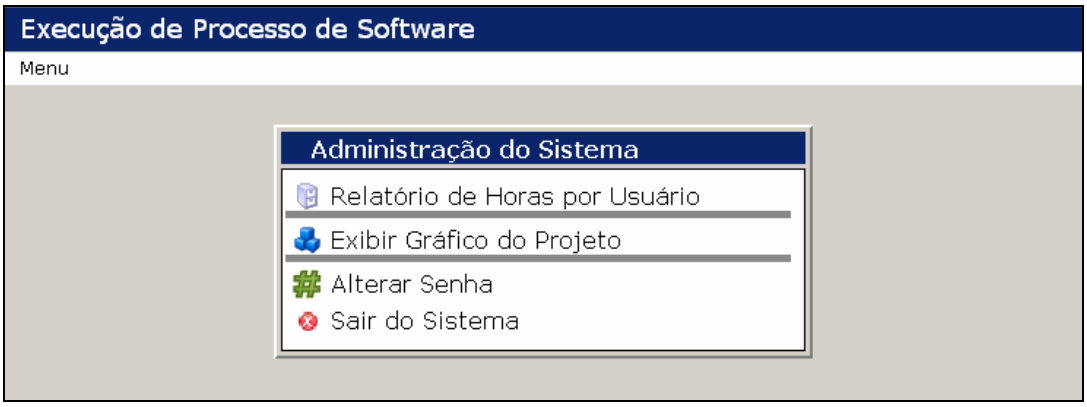


FIG 5.55 – Tela de opções do gerente após importação de dados.

Aos atores, não é dada a opção de Relatório de Horas por Usuário, mas é dada a opção de visualizar a sua agenda de tarefas como apresentado na FIG 5.56.

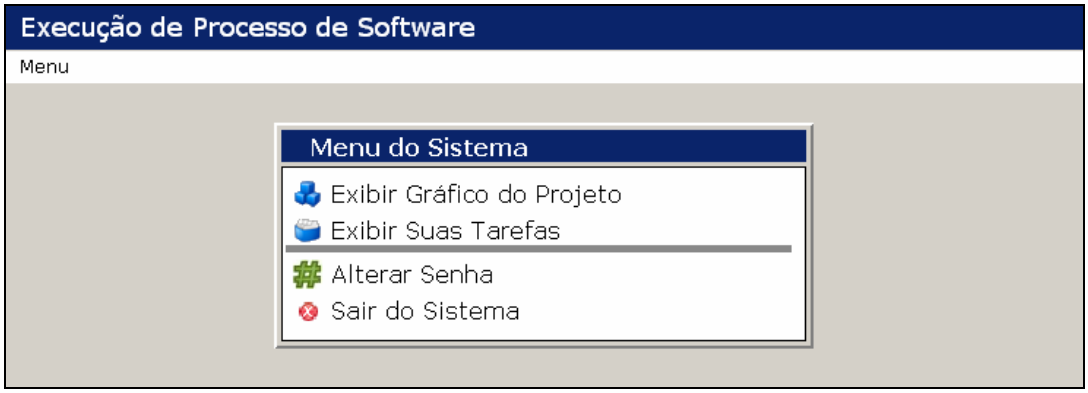


FIG 5.56 – Tela de opções do ator.

A agenda de tarefas do ator Thiago Cunha possui quatro tarefas onde cada uma se encontra em um estado diferente. A tarefa de Especificação de Características Funcionais se encontra concluída, a tarefa Definição de Entradas, Saídas e Interfaces se encontra atrasada e com 75% do andamento concluído, a tarefa Codificação e Teste está em andamento e 10% do andamento concluído e por fim a tarefa Elaboração do Manual do Sistema ainda não foi iniciada (FIG 5.57). Para a tarefa que se encontra atrasada, o ator recebe diariamente uma notificação por e-mail indicando o estado da mesma.

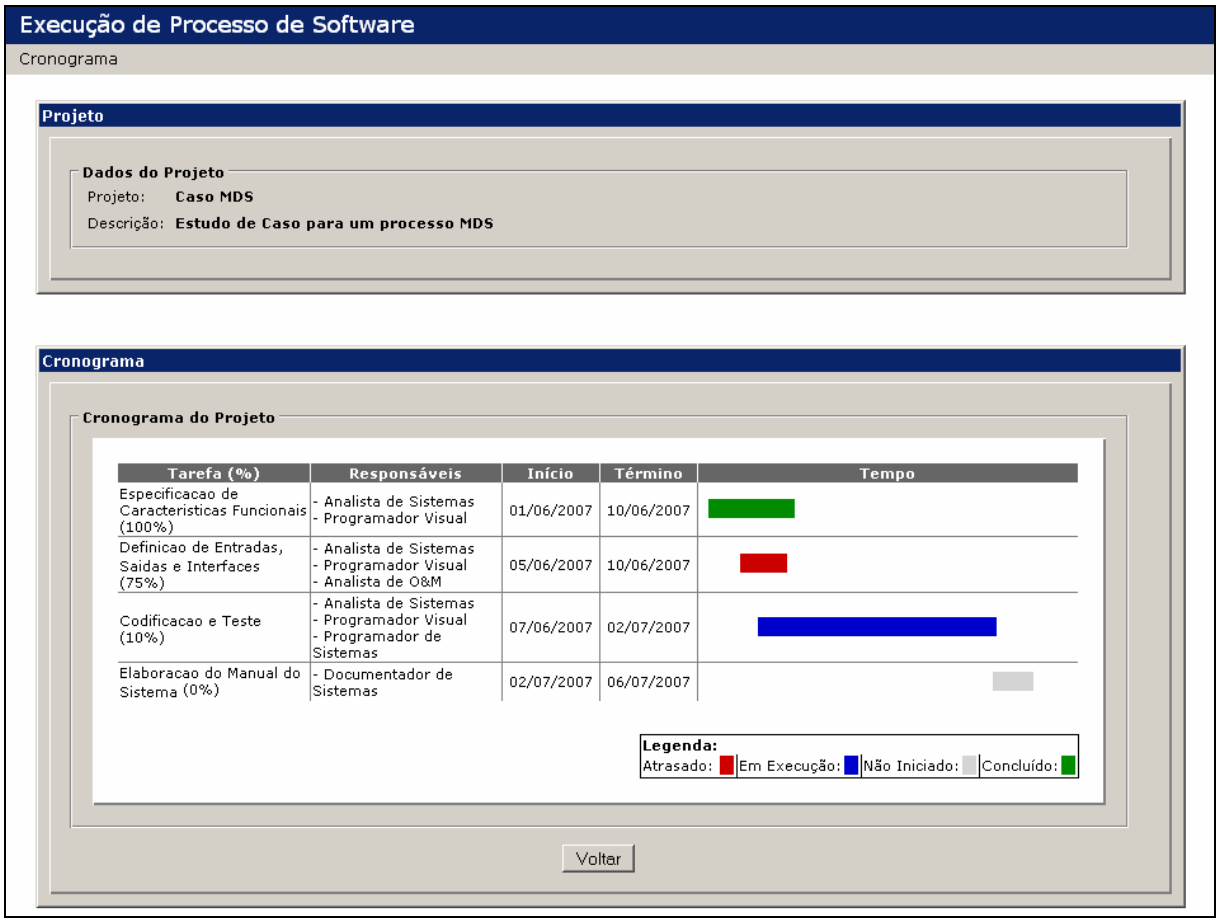


FIG 5.57 – Agenda do ator Thiago Cunha.

Assim como o gerente, os atores podem visualizar o estado e os detalhes de todas as tarefas do processo para que todos possam ter ciência do andamento do projeto. Clicando no nome da tarefa o usuário é direcionado para sua tela de detalhes, onde podem ser vistos os dados da sua descrição e os passos para sua execução (FIG 5.58).

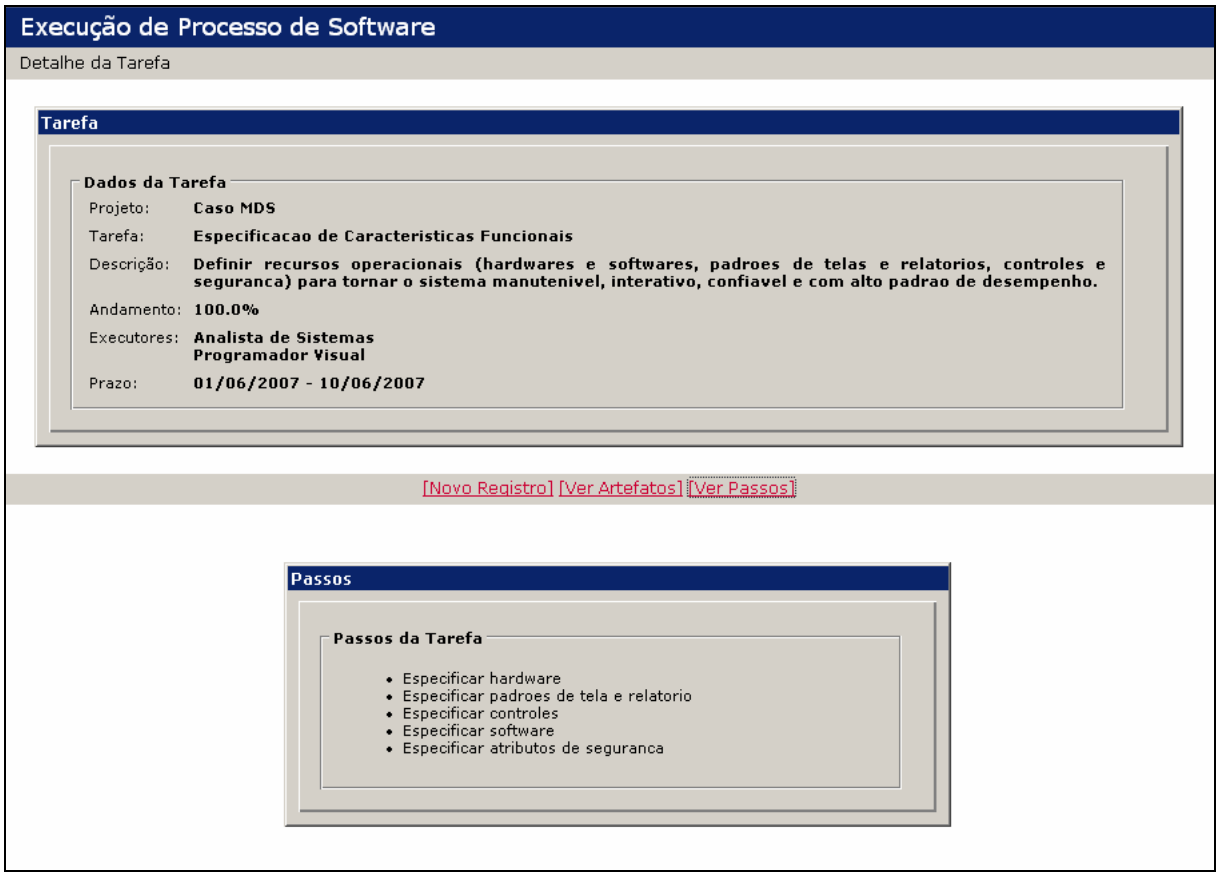


FIG 5.58 – Tela de detalhes da tarefa.

Pode-se ainda navegar nos detalhes, exibindo a lista de artefatos ou a lista de ocorrências. Cada artefato possui também uma tela de detalhes com os dados da sua modelagem e instanciação, onde é permitido ao usuário registrar uma nova versão para este artefato ou recuperar uma versão já registrada no sistema (FIG 5.59).

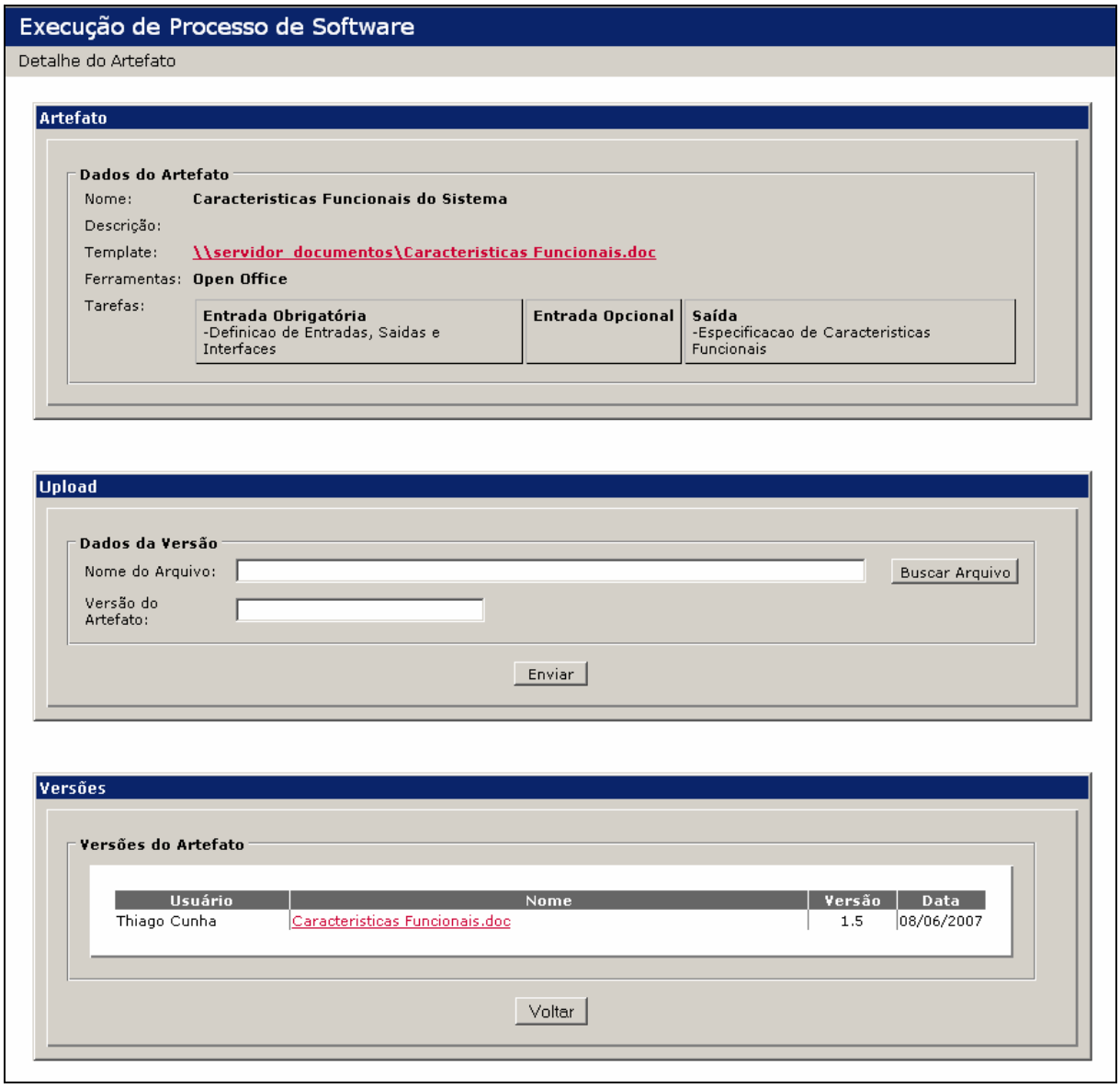


FIG 5.59 – Tela de detalhes do artefato.

Ao navegar pelas ocorrências da tarefa, o ator pode além de ler as ocorrências já registradas, registrar uma nova indicando uma data, tempo utilizado, porcentagem do andamento da tarefa e uma descrição do que foi feito. Um exemplo da listagem de ocorrências e formulário para criação de novos registros é exibido na FIG 5.60.

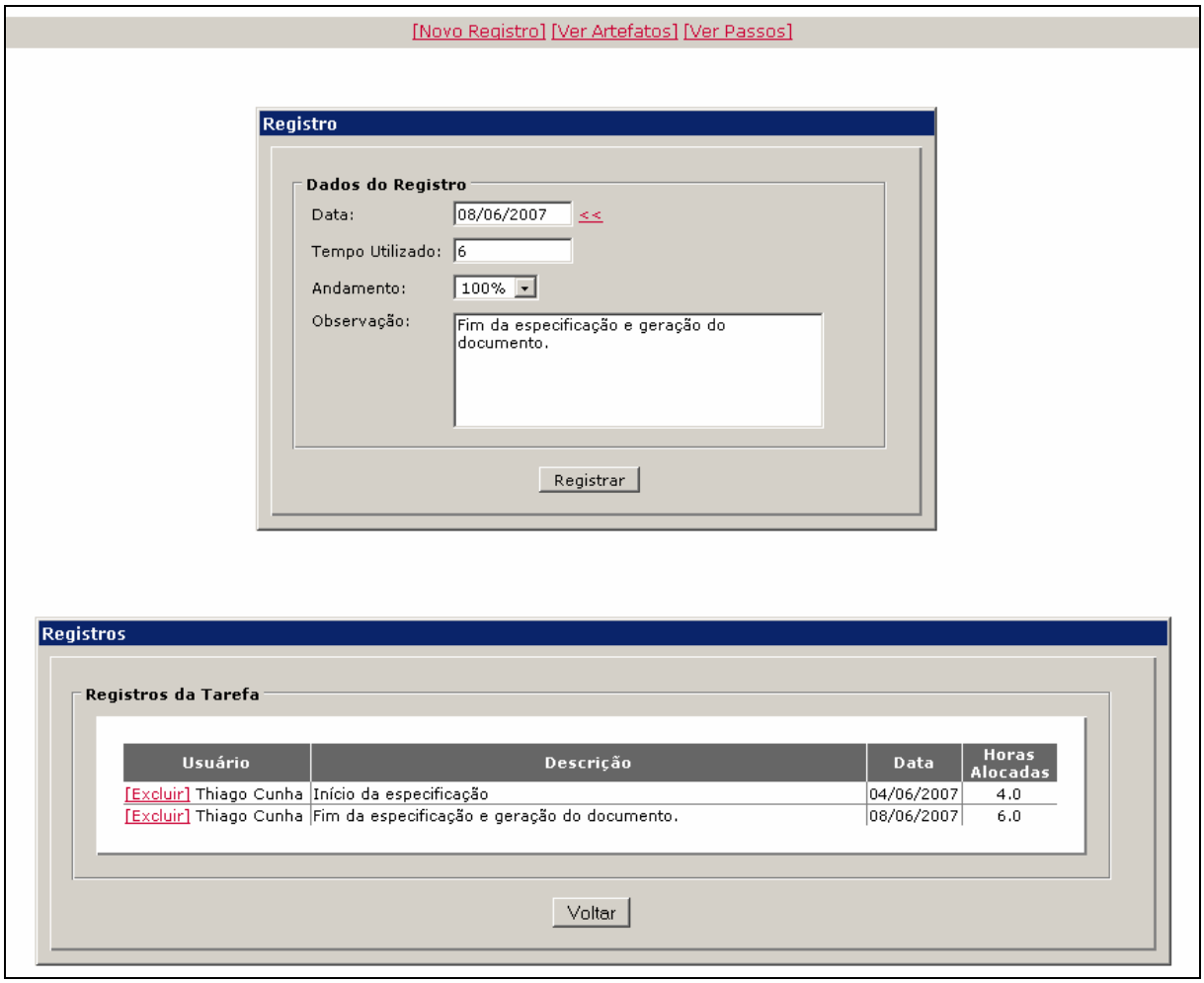


FIG 5.60 – Tela de registro de ocorrências.

Com estas informações registradas na lista de ocorrência, o gerente pode periodicamente consultar estas informações para detectar problemas ou eventuais atrasos. Nestes casos, ele pode optar por modificar dinamicamente, durante a execução do processo, alguns prazos ou alocar tarefas para outros papéis que não os definidos na modelagem reduzindo o impacto destes problemas ou atrasos.

5.2.4. LIÇÕES APRENDIDAS

Com a realização desta segunda prova de conceito, foi possível experimentar melhor as possíveis situações de atraso, notificações e registro de ocorrências. Conforme as necessidades de novas informações e funcionalidades do ambiente foram surgindo, estas foram implementadas e disponibilizadas com o objetivo de aumentar o controle sobre o processo em execução.

6. CONCLUSÃO

Ter um processo de desenvolvimento bem definido facilita o desenvolvimento de sistemas complexos. Em contrapartida, desenvolver sistemas de forma *ad hoc* acaba por comprometer o sucesso do projeto. Ter o processo de desenvolvimento definido permite que o mesmo seja analisado e como resultado desta análise, ele possa sofrer evoluções que o tornarão mais maduro e influenciarão na qualidade do produto final.

Este trabalho descreveu o Ambiente Integrado para Processos de Software (AIPS) como proposta de um ambiente integrado baseado em software de código aberto para modelagem, instanciação e execução de processos de desenvolvimento de software. A utilização de ambientes para modelagem e controle de processos de software traz benefícios, como melhor comunicação entre as pessoas envolvidas no desenvolvimento, suporte à reutilização de modelos de processo e o acompanhamento dos prazos e utilização de recursos. Mecanismos integrados permitem que mesmo mantendo estas aplicações separadas, a troca de informações entre elas ocorra de forma natural e transparente ao usuário.

A ferramenta de modelagem possibilita a definição de processos de desenvolvimento permitindo que sejam criados ou explicitados os padrões e processos utilizados no desenvolvimento de software. Possui recursos para definição dos elementos componentes do processo e a partir da composição destes elementos é possível estruturar todo o processo.

A proposta para instanciação permite que informações definidas na modelagem sejam estendidas e transformem o modelo abstrato do processo em um modelo executável, além de permitir que um mesmo modelo abstrato seja instanciado inúmeras vezes, garantindo que não haja necessidade de gerar um modelo para cada projeto onde o processo for aplicado. Nesta extensão de informações estão: o número de iterações para cada atividade existente, garantindo o suporte a processos iterativos; data de início e fim para cada uma das tarefas; nome e email dos atores que irão representar os papéis existentes no processo e o caminho para o *template* e as ferramentas que devem ser utilizadas e para geração de cada artefato.

A execução disponibilizará uma agenda ao ator indicando a lista de tarefas sob sua responsabilidade no projeto, permitirá a visualização de detalhes para estas tarefas e os artefatos utilizados e gerados pelas mesmas. Estes detalhamentos garantem que ator tenha acesso às informações necessárias para a execução de suas tarefas e com isso as expectativas do gerente sejam atendidas. Permite também ao ator registrar ocorrências para as tarefas com descrição dos sucessos ou problemas encontrados, horas gastas e porcentagem de conclusão,

que servirão como histórico para o gerente fazer projeções e, se achar necessário, alterar prazos ou realocar recursos para obter sucesso no projeto.

Espera-se que, com o uso do AIPS, um gerente de projetos consiga modelar, instanciar e executar um processo de software de forma integrada, além de acompanhar e controlar todo o processo estando sempre ciente dos acontecimentos mesmo sem estar fisicamente junto dos demais integrantes da equipe. Espera-se também que os atores do processo tenham acesso ao conjunto de informações necessárias para execução de suas tarefas e com isso atendam as expectativas do gerente em relação a prazos e produtos gerados.

6.1. CONTRIBUIÇÕES

As contribuições alcançadas por este trabalho foram:

Prover, de forma integrada um mecanismo de modelagem, instanciação e execução de processos de software, criando um ambiente integrado para a gerência de processos de desenvolvimento de software. O AIPS é um ambiente que integra as etapas de modelagem, instanciação e execução de processos. Com o ambiente trabalhando de forma integrada, a troca de informações entre as fases é feita sem a necessidade de qualquer adaptação. Esta integração é obtida com a utilização dos produtos gerados pelas fases de modelagem e instanciação para carregar os dados das fases de instanciação e execução respectivamente, sem que estes produtos necessitem de qualquer tratamento específico.

Oferecer um mecanismo de controle de execução de processos que disponibilize aos atores do processo informações detalhadas de suas tarefas. O mecanismo de execução do AIPS disponibiliza aos atores do processo detalhes importantes para a execução de suas tarefas como: data prevista de início e fim, descrição, passos a serem seguidos e artefatos de entrada e saída, que guiarão o ator ao longo do projeto. Para cada artefato associado a tarefa, é disponibilizado caminho de acesso ao *template* do documento e a ferramenta que deve ser utilizada para gerá-lo. O mecanismo ainda funciona como um repositório de documentos e permite que sejam acessadas e registradas versões para estes documentos facilitando o aceso e a distribuição destas informações. Permite também que sejam registradas ocorrências para cada uma das tarefas, onde se deve definir uma descrição para a ocorrência, tempo gasto e porcentagem de conclusão.

Desenvolver uma ferramenta que permita a gestão de processos de desenvolvimento. O AIPS permite ao gerente do projeto acompanhar detalhadamente o andamento das diversas tarefas do projeto, inspecionando as ocorrências e artefatos registrados. Com estas informações é possível ter uma visão do estado do projeto como um todo e o ajuda a prever possíveis problemas ou atrasos. Assim, durante a execução, ele pode realocar ou rever os prazos das tarefas com o objetivo de diminuir o impacto destes problemas. Estas informações servirão como um histórico de aprendizado para projetos futuros.

Desenvolver as ferramentas como software livre, para que estes possam sofrer evoluções por parte dos usuários e participantes da comunidade. O ambiente proposto neste trabalho foi desenvolvido utilizando apenas recursos de código aberto como o servidor de aplicações Apache TomCat, banco de dados MySQL, ambiente de desenvolvimento Eclipse além de linguagem e tecnologias Java. Todos os mecanismos criados e que fazem parte do ambiente, também serão disponibilizados sob a licença de software livre. Desta forma, as ferramentas poderão ser utilizadas para qualquer uso, poderão ser estudadas e adaptadas de acordo com a necessidade do usuário e poderão sofrer modificações de modo que toda a comunidade se beneficie com a sua melhoria.

6.2. TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, é apontada a disponibilização do arquivo de *template* do artefato, ao invés de apenas ser indicado o caminho de acesso a tal arquivo. Dessa forma, para os casos onde o caminho indicado é um endereço de diretório de rede, o *template* estaria acessível para qualquer pessoa que acessasse o mecanismo de execução independente do local onde estivesse realizando o acesso.

Uma outra sugestão seria o tratamento das restrições de dependências entre as tarefas, especificadas na modelagem. Com isso, um ator poderia, por exemplo, ser notificado automaticamente pelo sistema de que o artefato que aguardava ser gerado para dar início a sua tarefa já estaria disponível sem que o mesmo tivesse a necessidade de entrar constantemente no sistema para verificar esta disponibilidade. Isso também poderia evitar uma instanciação equivocada em relação às datas das tarefas, impedindo que uma tarefa que dependa do fim de outra, não se inicie antes que esta outra termine.

Mecanismos de coleta automática de métricas também poderiam ser implementados, uma vez que os dados de evolução de um processo são importantes para o controle e aperfeiçoamento do mesmo. Outros estudos de caso poderiam ser feitos no ambiente, experimentando novos tipos de processos e novas necessidades. Isto traria uma maior maturidade ao ambiente e cooperaria na identificação de novos requisitos e pontos de controle.

Poderia ser avaliada a possibilidade de instanciação e execução de processos modelados em outras ferramentas que não o EPF, mas que também utilizem a *Unified Method Architecture*, presente no meta-modelo do SPEM. Neste caso os mecanismos poderiam abranger um número maior de ferramentas de modelagem, dando ao usuário opções de escolha para a ferramenta que mais atende suas necessidades.

Outro estudo interessante seria a possibilidade de integração de ferramentas que suportam o processo de desenvolvimento com o ambiente criado, que poderia agregar questões interessantes. Por exemplo, uma integração da interface de desenvolvimento com o mecanismo de execução poderá de alguma forma induzir o ator a registrar o trabalho realizado por ele, garantindo que o estado do projeto no mecanismo de execução reflita o estado real.

REFERÊNCIAS BIBLIOGRÁFICAS

ACUÑA, SILVIA T.; DE ANTONIO A., FERRÉ X., MATÉ L., LÓPEZ M.: **The Software Process: Modelling, Evaluation and Improvement.** Argentina: World Scientific Publishing Company, 2000.

ALLILAIRE, F.; IDRISSI, T. **ADT: Eclipse Development Tools for ATL.** In Proceedings of the Second European Workshop on Model Driven Architecture (EWMDA-2), pages 171-178, 2004.

ARMENISE, P.; BANDINELLI, S. GHEZZI, C.; MORZENTI, A. **Software Processes Representation Languages: Survey and Assessment.** In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 4., 1992, Capri, Italy. Pages 455-462. IEEE Press, June 1992.

BALDUINO, R.; LYONS, B.: **OpenUP/Basic - A Process for Small and Agile Projects. 2006.** Disponível em http://www.eclipse.org/epf/general/OpenUP_Basic.pdf. Último acesso em 20/03/2007.

BERGER, P. M., **Instanciação de Processos de Software em Ambientes Configurados na Estação TABA**, Dissertação de Mestrado, COPPE, Universidade Federal do Rio de Janeiro, 2003. 128 páginas.

CARNIELLO, Andréia, Itana M. S. Gimenes, **Um ambiente de programação de processos cooperativos de software para o ExPSEE**, Universidade Estadual de Maringá, Paraná, 1999.

CUGOLA, G.; GHEZZI, C. **Software Processes: a Retrospective and a Path to the Future.** Software process - Improvement and practice, vol. 4, pp. 101-123, 1998.

CURTIS, B.; KELLNER, Marc I.; OVER, J.: **Process modeling, Communications of the ACM**, v.35 n.9, p.75-90, Setembro. 1992.

DAL MORO, R; NARDI, J.C.; FALBO, R DE ALMEIDA. **ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software.** XII Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software, SBES'2005. Uberlândia, Brasil, Outubro 2005.

DERNIAME, J.; KABA, B.; WASTELL, D. **Software Process: Principles, Methodology and Technology.** , Lecture Notes in Computer Science, Vol. 1500, Springer Verlag 1999, pp. 53-93.

DEVX. **The Eclipse Process Framework Puts Knowledge at Users' Fingertips.** Disponível em http://www.devx.com/ibm/Article/30221. Último acesso em 20/03/2007.

ECLIPSE. **Eclipse Project.** Disponível em http://www.eclipse.org/. Último acesso em 25/10/2006.

EPF. **Eclipse Process Framework.** Disponível em http://www.eclipse.org/epf/. Último acesso em 25/10/2006.

ESTOLANO, Mário Henrique da Rocha**. Base de Métricas para a Estação TABA**. Dissertação de Mestrado COPPE/UFRJ. Rio de Janeiro, 2005. 133 páginas.

FANTINATO, M. **Um Escalonador de Tarefas para o ExPSEE.** Universidade Estadual de Maringá, Paraná, 1999.

FEILER, P.; HUMPHREY, W. **Software Process Development and Enactment: Concepts and Definitions.** In Proceedings of the 2nd International Conference on Software Process. IEEE Computer Society Press, Los Alamitos, CA. 1993.

FRANÇA, B.; REIS, C. **Documentação de Referência do Sistema WebAPSEE.** Versão 1.0. 2007b. Disponível em http://labes.ufpa.br:8080/labes/CD_Card/Interna/Documentacao.html. Último acesso em 20/03/2007.

FUGGETTA, A. **Software Process: A Roadmap. In: Finkelstein (Ed.), Future of Software Engineering.** In Anthony Finkelstein, ed. The future of Software Engineering. ACM Press, pp27-34, 2000

GENVIGIR, ELIAS C.; FILHO, LUIZ F. B.; SANT'ANNA, NILSON. **Modelagem de Processos de Software Através do SPEM - Software Process Engineering Metamodel - Conceitos e Aplicação.** III WORCAP, São Jose dos Campos, SP. Novembro, 2003.

GIMENES, I.M. **Uma Introdução ao Processo de Engenharia de Software: Ambientes e Formalismos,** XIII Jornada de Atualização em Informática, Caxambu - MG , agosto, 1994.

HAUMER, PETER. **Eclipse Process Framework Composer Overview.** Disponível em http://www.eclipse.org/epf/general/getting_started.php. Último acesso 20/03/2007.

HUFF, K. **Software process modeling**. In Trends in Software: Software Process, Fuggetta, A. and Wolf, A., Eds. John Wiley and Sons Ltd., Chichester, UK., 1996.

KRUCHTEN, P., **The Rational Unified Process – An Introduction**, Addison-Wesley, Boston, USA. 1999.

LIMA, A.; FRANÇA, B. B. N.; Lima REIS, Carla Alessandra; REIS, Rodrigo Quites; COSTA, A. J. S: **Gerência Flexível de Processos de Software com o Ambiente WebAPSEE.** In: Salão de Ferramentas - XX SBES, 2006, Florianópolis, SC. Simpósio Brasileiro de Engenharia de Software, 2006.

LIMA, A.: **WebAPSEE: Um Ambiente Livre e Flexível para Gerência de Processos de Software**. VII Workshop de Software Livre. Porto Alegre: Abril/2006.

LIMA, C. A. G., REIS, R. Q., NUNES, D. J. **Gerenciamento do processo de desenvolvimento cooperativo de software no ambiente PROSOFT.** In: Simpósio Brasileiro de Engenharia de Software, 12., 1998, Maringá - Paraná. Maringá: Ideal Ind. Gráf., 1998.

LONCHAMP, J. **A Structured Conceptual and Terminological Framework for Software Process Engineering.** In Second International Conference on the Software Process - Continuous Software Improvement, pages 41 - 53, Los Alamitos, California, 1993. IEEE Computer Society Press.

MOITRA, D. **India's Software Industry: the software superpower.** IEEE Software, Vol. 18, Issue 1, p. 77-80, January/February, 2001.

OLIVEIRA JUNIOR, E. A.; GIMENES, I. M.: **Especificação de um Sistema Gerenciador de Workflow de Acordo com a Abordagem de Desenvolvimento Baseado em Componentes.** Revista Eletrônica de Iniciação Científica, Porto Alegre-RS, v. 3, n. 1, 2003.

OSTERWEIL, L. **Software Processes are Software Too.** In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 1987, Monterey, California, USA. Los Alamitos: IEEE Press, 1987. p. 2-13.

PRESSMAN, R. S.: **Software Engineering**, MacGraw Hill, 4ª edition, New York, NY, 1997.

PRODERJ: **Método de Desenvolvimento de Sistemas**, Versão 3.0. Fevereiro/2006

RABELO, A.; REIS, Carla A.L.; REIS, Rodrigo Q.; PIMENTA, Marcelo S.; NUNES Daltro J.: **Analisando a Interação de Gerentes e Desenvolvedores em Ambientes de Processo de Software: Classificação e Exemplos.** In: I Workshop Chileno de Engenharia de Software, 2001, Punta Arenas, 2001.

REIS, C. A. L. **Um Modelo Flexível para Execução de Processos de Software.** Porto Alegre. PPGC-UFRGS. 2003.

REIS, C. A. L. **Uma Abordagem Flexível para Execução de Processos de Software Evolutivos.** Tese de Doutorado. UFRGS - PPGC. 267 páginas. Maio de 2003.

REIS, R. Q. **Uma Avaliação dos Paradigmas de Linguagens de Processo de Software.** Trabalho Individual II, Porto Alegre: PPGC-UFRGS, 49 páginas, Março 1999.

ROCHA, A.R. C., MALDONADO, J.C., WEBER, K. C. **Qualidade de Software**. São Paulo. Prentice Hall. 2001.

SILVA, F. A. D. D. **Um Modelo de Simulação de Processos de Software Baseado em Conhecimento para o Ambiente PROSOFT.** Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 124 páginas. 2001.

SILVA, RAC; SOARES, LS; BRAGA , JL. **Workflow Aplicado a Engenharia de Software Baseada em Processos: Uma Visão Geral.** INFOCOMP. Journal of Computer Science, v. 5, p. 76-84, 2006

SILVA FILHO, R. C. **Uma Abordagem para Avaliação de Propostas de Melhoria em Processos de Software**. Dissertação de Mestrado COPPE/UFRJ. Rio de Janeiro, 2006. 178 páginas.

SOUZA, ABRAHAM L. R., REIS, C. A. L.: **Interação Humana Durante Execução de Processos de Software: Classificação e Exemplos**. Relatório de Pesquisa RP 310. Porto Alegre: PPGC-UFRGS, p. 473-492, Jun. 2001.

SPEM, **Software Process Engineering Metamodel Specification.** Versão 1.1. Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. 2005.

SUTTON, S.; OSTERWEIL, L. **An Analysis of Process Languages.** Technical Report 95 78, Dept. of Computer Science, University of Massachusetts, Amherst, 25 pages, 1995.

TULLY, C. **Representing and Enacting the Software Process** of the 4th International Software Process Workshop, ACM SlGSOFT Software Engineering Notes, ACM Press, June 1989.

7. ANEXOS

7.1. ANEXO A

PRODUTO DA FASE DE MODELAGEM - PROCESSO EXEMPLO

```
<?xml version="1.0" encoding="UTF-8"?>
<uma:MethodLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:uma="http://www.eclipse.org/epf/uma/1.0.3" epf:version="1.0.0"
   name="SPEM Example" briefDescription="" id="_aVl6UAilEdyJg53g-AgF8w"
   orderingGuide="" suppressed="false" authors="" changeDescription=""
   version="">
  <MethodPlugin name="SPEM Example" briefDescription=""
   id="_srD1wAilEdyJg53g-AgF8w" orderingGuide="" suppressed="false"
   authors="" changeDescription="" version="" userChangeable="true">
    <MethodPackage xsi:type="uma:ContentCategoryPackage"
   name="ContentCategories" id="_Th8zYA1zEdy7Ptr6UpZ43g"/>
    <MethodPackage xsi:type="uma:ContentPackage" name="My Package"
   briefDescription="" id="_vWuKwAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" global="false">
      <ContentElement xsi:type="uma:Role" name="Functional Analyst"
   briefDescription="" id="_xUCzEAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Functional Analyst"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Role" name="Interface Designer"
   briefDescription="" id="_0bj9oAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Interface Designer"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Role" name="Technical Designer"
   briefDescription="" id="_2mmR0AilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Technical Designer"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Requirements"
   briefDescription="" id="_5f9roAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Requirements"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Work Processes"
   briefDescription="" id="_7LQIwAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Work Processes"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Interface {draft}"
   briefDescription="" id="_9eLNoAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Interface {draft}"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Interface
   {refined}" briefDescription="" id="_-xxjgAilEdyJg53g-AgF8w"
   orderingGuide="" suppressed="false" presentationName="User Interface
   {refined}" variabilityType="na"/>
      <ContentElement xsi:type="uma:Task" name="Define Requirements"
   briefDescription="" id="_Ag70AAimEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Define Requirements"
   variabilityType="na">
        <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
        <Output>_5f9roAilEdyJg53g-AgF8w</Output>
      </ContentElement>
      <ContentElement xsi:type="uma:Task" name="Design Process Model"
   briefDescription="" id="_HAOysAimEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Design Process Model"
   variabilityType="na">
        <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
```

```
     <Output>_7LQIwAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Draft User Interface"
briefDescription="" id="_MLyRkAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Draft User Interface"
variabilityType="na">
     <PerformedBy>_0bj9oAilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_5f9roAilEdyJg53g-AgF8w</MandatoryInput>
     <Output>_9eLNoAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Define Tech. Requirements"
briefDescription="" id="_RE25UAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Define Tech. Requirements"
variabilityType="na">
     <PerformedBy>_2mmR0AilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_9eLNoAilEdyJg53g-AgF8w</MandatoryInput>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Refine User Interface"
briefDescription="" id="_UWN8YAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Refine User Interface"
variabilityType="na">
     <PerformedBy>_0bj9oAilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_7LQIwAilEdyJg53g-AgF8w</MandatoryInput>
     <Output>_-xxjgAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Build Application"
briefDescription="" id="_ZzNJgAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Build Application"
variabilityType="na">
     <PerformedBy>_2mmR0AilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_-xxjgAilEdyJg53g-AgF8w</MandatoryInput>
   </ContentElement>
 </MethodPackage>
 <MethodPackage xsi:type="uma:ProcessComponent" name="Example"
briefDescription="" id="_jDrPUAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" global="false" authors="" changeDescription=""
version="">
   <Process xsi:type="uma:DeliveryProcess" name="Example"
briefDescription="" id="_jDrPUQimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Example"
hasMultipleOccurrences="false" isOptional="false" isPlanned="true"
prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false"
IsEnactable="false" variabilityType="na">
     <Presentation xsi:type="uma:DeliveryProcessDescription"
name="Example,_jDrPUQimEdyJg53g-AgF8w" briefDescription="" id="-cX-
vdSTwcrQkGTo9eLDIdw" orderingGuide="" suppressed="false" authors=""
changeDescription="" version="1.0.0" usageGuidance="" externalId="">
       <MainDescription></MainDescription>
       <KeyConsiderations></KeyConsiderations>
       <Alternatives></Alternatives>
       <HowToStaff></HowToStaff>
       <Purpose></Purpose>
       <Scope></Scope>
       <UsageNotes></UsageNotes>
       <Scale></Scale>
       <ProjectCharacteristics></ProjectCharacteristics>
       <RiskLevel></RiskLevel>
       <EstimatingTechnique></EstimatingTechnique>
       <ProjectMemberExpertise></ProjectMemberExpertise>
       <TypeOfContract></TypeOfContract>
```

```
     </Presentation>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Define
Requirements" briefDescription="" id="_xDFJcAimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Define
Requirements" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Task>_Ag70AAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <Output>_mgewkQiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Design
Process Model" briefDescription="" id="_9C8NsAimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Design Process
Model" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="__EufRAimEdyJg53g-
AgF8w">_xDFJcAimEdyJg53g-AgF8w</Predecessor>
       <Task>_HAOysAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <Output>_rEGFsgiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Draft User
Interface" briefDescription="" id="_97at0AimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Draft User
Interface" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="__ofTpAimEdyJg53g-
AgF8w">_xDFJcAimEdyJg53g-AgF8w</Predecessor>
       <Task>_MLyRkAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_mgewkQiqEdyTutl_HeDilg</MandatoryInput>
       <Output>_rEGFswiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Define Tech.
Requirements" briefDescription="" id="_Fz9-cAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Define Tech.
Requirements" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_HS4zhAinEdyJg53g-
AgF8w">_97at0AimEdyJg53g-AgF8w</Predecessor>
       <Task>_RE25UAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsQiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFswiqEdyTutl_HeDilg</MandatoryInput>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Refine User
Interface" briefDescription="" id="_DpPMQAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Refine User
Interface" hasMultipleOccurrences="false" isOptional="false"
```

```
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_R-HAEQinEdyJg53g-
AgF8w">_9C8NsAimEdyJg53g-AgF8w</Predecessor>
       <Predecessor linkType="finishToFinish" id="_R-HAEginEdyJg53g-
AgF8w">_Fz9-cAinEdyJg53g-AgF8w</Predecessor>
       <Task>_UWN8YAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFsgiqEdyTutl_HeDilg</MandatoryInput>
       <Output>_rEGFtAiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Build
Application" briefDescription="" id="_hJgzQAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Build Application"
hasMultipleOccurrences="false" isOptional="false" isPlanned="false"
prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false"
isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_i0qGdAinEdyJg53g-
AgF8w">_DpPMQAinEdyJg53g-AgF8w</Predecessor>
       <Task>_ZzNJgAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsQiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFtAiqEdyTutl_HeDilg</MandatoryInput>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Functional
Analyst" briefDescription="" id="_mgewkAiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Functional
Analyst" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_xUCzEAilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
Requirements" briefDescription="" id="_mgewkQiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="User Requirements"
hasMultipleOccurrences="false" isOptional="false" isPlanned="true"
prefix="" isSynchronizedWithSource="true" activityEntryState=""
activityExitState="">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <WorkProduct>_5f9roAilEdyJg53g-AgF8w</WorkProduct>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Interface
Designer" briefDescription="" id="_rEGFsAiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Interface
Designer" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_0bj9oAilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Technical
Designer" briefDescription="" id="_rEGFsQiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Technical
Designer" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_2mmR0AilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
```

```
        <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
    Work Processes" briefDescription="" id="_rEGFsgiqEdyTutl_HeDilg"
    orderingGuide="" suppressed="false" presentationName="User Work
    Processes" hasMultipleOccurrences="false" isOptional="false"
    isPlanned="true" prefix="" isSynchronizedWithSource="true"
    activityEntryState="" activityExitState="">
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <WorkProduct>_7LQIwAilEdyJg53g-AgF8w</WorkProduct>
        </BreakdownElement>
        <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
    Interface {draft}" briefDescription="" id="_rEGFswiqEdyTutl_HeDilg"
    orderingGuide="" suppressed="false" presentationName="User Interface
    {draft}" hasMultipleOccurrences="false" isOptional="false"
    isPlanned="true" prefix="" isSynchronizedWithSource="true"
    activityEntryState="" activityExitState="">
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <WorkProduct>_9eLNoAilEdyJg53g-AgF8w</WorkProduct>
        </BreakdownElement>
        <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
    Interface {refined}" briefDescription="" id="_rEGFtAiqEdyTutl_HeDilg"
    orderingGuide="" suppressed="false" presentationName="User Interface
    {refined}" hasMultipleOccurrences="false" isOptional="false"
    isPlanned="true" prefix="" isSynchronizedWithSource="true"
    activityEntryState="" activityExitState="">
          <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
          <WorkProduct>_-xxjgAilEdyJg53g-AgF8w</WorkProduct>
        </BreakdownElement>
        <DefaultContext>_t_stcAilEdyJg53g-AgF8w</DefaultContext>
        <ValidContext>_t_stcAilEdyJg53g-AgF8w</ValidContext>
      </Process>
    </MethodPackage>
  </MethodPlugin>
  <MethodConfiguration name="SPEM Example" briefDescription=""
    id="_t_stcAilEdyJg53g-AgF8w" orderingGuide="" suppressed="false"
    authors="" changeDescription="" version="1.0.0">
    <MethodPluginSelection>_srD1wAilEdyJg53g-AgF8w</MethodPluginSelection>
    <MethodPackageSelection>_srD1zAilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1wQilEdyJg53g-
    AgF8w</MethodPackageSelection>

    <MethodPackageSelection>_Th8zYA1zEdy7Ptr6UpZ43g</MethodPackageSelection>
    <MethodPackageSelection>_srD1wgilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1zQilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1zgilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1zwilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1yAilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_srD1ygilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_vWuKwAilEdyJg53g-
    AgF8w</MethodPackageSelection>
    <MethodPackageSelection>_jDrPUAimEdyJg53g-
    AgF8w</MethodPackageSelection>
  </MethodConfiguration>
</uma:MethodLibrary>
```

7.2. ANEXO B

PRODUTO DA FASE DE INSTANCIAÇÃO - PROCESSO EXEMPLO

```
<?xml version="1.0" encoding="UTF-8"?>
<uma:MethodLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:uma="http://www.eclipse.org/epf/uma/1.0.3" epf:version="1.0.0"
   name="SPEM Example" briefDescription="" id="_aVl6UAilEdyJg53g-AgF8w"
   orderingGuide="" suppressed="false" authors="" changeDescription=""
   version="">
  <MethodPlugin name="SPEM Example" briefDescription=""
   id="_srD1wAilEdyJg53g-AgF8w" orderingGuide="" suppressed="false"
   authors="" changeDescription="" version="" userChangeable="true">
    <MethodPackage xsi:type="uma:ContentCategoryPackage"
   name="ContentCategories" id="_Th8zYA1zEdy7Ptr6UpZ43g"/>
    <MethodPackage xsi:type="uma:ContentPackage" name="My Package"
   briefDescription="" id="_vWuKwAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" global="false">
      <ContentElement xsi:type="uma:Role" name="Functional Analyst"
   briefDescription="" id="_xUCzEAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Functional Analyst"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Role" name="Interface Designer"
   briefDescription="" id="_0bj9oAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Interface Designer"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Role" name="Technical Designer"
   briefDescription="" id="_2mmR0AilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Technical Designer"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Requirements"
   briefDescription="" id="_5f9roAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Requirements"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Work Processes"
   briefDescription="" id="_7LQIwAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Work Processes"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Interface {draft}"
   briefDescription="" id="_9eLNoAilEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="User Interface {draft}"
   variabilityType="na"/>
      <ContentElement xsi:type="uma:Artifact" name="User Interface
   {refined}" briefDescription="" id="_-xxjgAilEdyJg53g-AgF8w"
   orderingGuide="" suppressed="false" presentationName="User Interface
   {refined}" variabilityType="na"/>
      <ContentElement xsi:type="uma:Task" name="Define Requirements"
   briefDescription="" id="_Ag70AAimEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Define Requirements"
   variabilityType="na">
        <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
        <Output>_5f9roAilEdyJg53g-AgF8w</Output>
      </ContentElement>
      <ContentElement xsi:type="uma:Task" name="Design Process Model"
   briefDescription="" id="_HAOysAimEdyJg53g-AgF8w" orderingGuide=""
   suppressed="false" presentationName="Design Process Model"
   variabilityType="na">
        <PerformedBy>_xUCzEAilEdyJg53g-AgF8w</PerformedBy>
```

```
     <Output>_7LQIwAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Draft User Interface"
briefDescription="" id="_MLyRkAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Draft User Interface"
variabilityType="na">
     <PerformedBy>_0bj9oAilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_5f9roAilEdyJg53g-AgF8w</MandatoryInput>
     <Output>_9eLNoAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Define Tech. Requirements"
briefDescription="" id="_RE25UAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Define Tech. Requirements"
variabilityType="na">
     <PerformedBy>_2mmR0AilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_9eLNoAilEdyJg53g-AgF8w</MandatoryInput>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Refine User Interface"
briefDescription="" id="_UWN8YAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Refine User Interface"
variabilityType="na">
     <PerformedBy>_0bj9oAilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_7LQIwAilEdyJg53g-AgF8w</MandatoryInput>
     <Output>_-xxjgAilEdyJg53g-AgF8w</Output>
   </ContentElement>
   <ContentElement xsi:type="uma:Task" name="Build Application"
briefDescription="" id="_ZzNJgAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Build Application"
variabilityType="na">
     <PerformedBy>_2mmR0AilEdyJg53g-AgF8w</PerformedBy>
     <MandatoryInput>_-xxjgAilEdyJg53g-AgF8w</MandatoryInput>
   </ContentElement>
 </MethodPackage>
 <MethodPackage xsi:type="uma:ProcessComponent" name="Example"
briefDescription="" id="_jDrPUAimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" global="false" authors="" changeDescription=""
version="">
   <Process xsi:type="uma:DeliveryProcess" name="Example"
briefDescription="" id="_jDrPUQimEdyJg53g-AgF8w" orderingGuide=""
suppressed="false" presentationName="Example"
hasMultipleOccurrences="false" isOptional="false" isPlanned="true"
prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false"
IsEnactable="false" variabilityType="na">
     <Presentation xsi:type="uma:DeliveryProcessDescription"
name="Example,_jDrPUQimEdyJg53g-AgF8w" briefDescription="" id="-cX-
vdSTwcrQkGTo9eLDIdw" orderingGuide="" suppressed="false" authors=""
changeDescription="" version="1.0.0" usageGuidance="" externalId="">
       <MainDescription></MainDescription>
       <KeyConsiderations></KeyConsiderations>
       <Alternatives></Alternatives>
       <HowToStaff></HowToStaff>
       <Purpose></Purpose>
       <Scope></Scope>
       <UsageNotes></UsageNotes>
       <Scale></Scale>
       <ProjectCharacteristics></ProjectCharacteristics>
       <RiskLevel></RiskLevel>
       <EstimatingTechnique></EstimatingTechnique>
       <ProjectMemberExpertise></ProjectMemberExpertise>
       <TypeOfContract></TypeOfContract>
```

```
     </Presentation>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Define
Requirements" briefDescription="" id="_xDFJcAimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Define
Requirements" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Task>_Ag70AAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <Output>_mgewkQiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Design
Process Model" briefDescription="" id="_9C8NsAimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Design Process
Model" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="__EufRAimEdyJg53g-
AgF8w">_xDFJcAimEdyJg53g-AgF8w</Predecessor>
       <Task>_HAOysAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_mgewkAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <Output>_rEGFsgiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Draft User
Interface" briefDescription="" id="_97at0AimEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Draft User
Interface" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="__ofTpAimEdyJg53g-
AgF8w">_xDFJcAimEdyJg53g-AgF8w</Predecessor>
       <Task>_MLyRkAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_mgewkQiqEdyTutl_HeDilg</MandatoryInput>
       <Output>_rEGFswiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Define Tech.
Requirements" briefDescription="" id="_Fz9-cAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Define Tech.
Requirements" hasMultipleOccurrences="false" isOptional="false"
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_HS4zhAinEdyJg53g-
AgF8w">_97at0AimEdyJg53g-AgF8w</Predecessor>
       <Task>_RE25UAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsQiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFswiqEdyTutl_HeDilg</MandatoryInput>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Refine User
Interface" briefDescription="" id="_DpPMQAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Refine User
Interface" hasMultipleOccurrences="false" isOptional="false"
```

```
isPlanned="false" prefix="" isEventDriven="false" isOngoing="false"
isRepeatable="false" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_R-HAEQinEdyJg53g-
AgF8w">_9C8NsAimEdyJg53g-AgF8w</Predecessor>
       <Predecessor linkType="finishToFinish" id="_R-HAEginEdyJg53g-
AgF8w">_Fz9-cAinEdyJg53g-AgF8w</Predecessor>
       <Task>_UWN8YAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsAiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFsgiqEdyTutl_HeDilg</MandatoryInput>
       <Output>_rEGFtAiqEdyTutl_HeDilg</Output>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:TaskDescriptor" name="Build
Application" briefDescription="" id="_hJgzQAinEdyJg53g-AgF8w"
orderingGuide="" suppressed="false" presentationName="Build Application"
hasMultipleOccurrences="false" isOptional="false" isPlanned="false"
prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false"
isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Predecessor linkType="finishToFinish" id="_i0qGdAinEdyJg53g-
AgF8w">_DpPMQAinEdyJg53g-AgF8w</Predecessor>
       <Task>_ZzNJgAimEdyJg53g-AgF8w</Task>

<PerformedPrimarilyBy>_rEGFsQiqEdyTutl_HeDilg</PerformedPrimarilyBy>
       <MandatoryInput>_rEGFtAiqEdyTutl_HeDilg</MandatoryInput>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Functional
Analyst" briefDescription="" id="_mgewkAiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Functional
Analyst" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_xUCzEAilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
Requirements" briefDescription="" id="_mgewkQiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="User Requirements"
hasMultipleOccurrences="false" isOptional="false" isPlanned="true"
prefix="" isSynchronizedWithSource="true" activityEntryState=""
activityExitState="">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <WorkProduct>_5f9roAilEdyJg53g-AgF8w</WorkProduct>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Interface
Designer" briefDescription="" id="_rEGFsAiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Interface
Designer" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_0bj9oAilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
     <BreakdownElement xsi:type="uma:RoleDescriptor" name="Technical
Designer" briefDescription="" id="_rEGFsQiqEdyTutl_HeDilg"
orderingGuide="" suppressed="false" presentationName="Technical
Designer" hasMultipleOccurrences="false" isOptional="false"
isPlanned="true" prefix="" isSynchronizedWithSource="true">
       <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
       <Role>_2mmR0AilEdyJg53g-AgF8w</Role>
     </BreakdownElement>
```

```
      <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
 Work Processes" briefDescription="" id="_rEGFsgiqEdyTutl_HeDilg"
 orderingGuide="" suppressed="false" presentationName="User Work
 Processes" hasMultipleOccurrences="false" isOptional="false"
 isPlanned="true" prefix="" isSynchronizedWithSource="true"
 activityEntryState="" activityExitState="">
        <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
        <WorkProduct>_7LQIwAilEdyJg53g-AgF8w</WorkProduct>
      </BreakdownElement>
      <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
 Interface {draft}" briefDescription="" id="_rEGFswiqEdyTutl_HeDilg"
 orderingGuide="" suppressed="false" presentationName="User Interface
 {draft}" hasMultipleOccurrences="false" isOptional="false"
 isPlanned="true" prefix="" isSynchronizedWithSource="true"
 activityEntryState="" activityExitState="">
        <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
        <WorkProduct>_9eLNoAilEdyJg53g-AgF8w</WorkProduct>
      </BreakdownElement>
      <BreakdownElement xsi:type="uma:WorkProductDescriptor" name="User
 Interface {refined}" briefDescription="" id="_rEGFtAiqEdyTutl_HeDilg"
 orderingGuide="" suppressed="false" presentationName="User Interface
 {refined}" hasMultipleOccurrences="false" isOptional="false"
 isPlanned="true" prefix="" isSynchronizedWithSource="true"
 activityEntryState="" activityExitState="">
        <SuperActivity>_jDrPUQimEdyJg53g-AgF8w</SuperActivity>
        <WorkProduct>_-xxjgAilEdyJg53g-AgF8w</WorkProduct>
      </BreakdownElement>
      <DefaultContext>_t_stcAilEdyJg53g-AgF8w</DefaultContext>
      <ValidContext>_t_stcAilEdyJg53g-AgF8w</ValidContext>
    </Process>
  </MethodPackage>
</MethodPlugin>
<MethodConfiguration name="SPEM Example" briefDescription=""
 id="_t_stcAilEdyJg53g-AgF8w" orderingGuide="" suppressed="false"
 authors="" changeDescription="" version="1.0.0">
  <MethodPluginSelection>_srD1wAilEdyJg53g-AgF8w</MethodPluginSelection>
  <MethodPackageSelection>_srD1zAilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1wQilEdyJg53g-
 AgF8w</MethodPackageSelection>

 <MethodPackageSelection>_Th8zYA1zEdy7Ptr6UpZ43g</MethodPackageSelection>
  <MethodPackageSelection>_srD1wgilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1zQilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1zgilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1zwilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1yAilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_srD1ygilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_vWuKwAilEdyJg53g-
 AgF8w</MethodPackageSelection>
  <MethodPackageSelection>_jDrPUAimEdyJg53g-
 AgF8w</MethodPackageSelection>
</MethodConfiguration>
<Instancia>
```

```
    <Actor name="JoÃ£o Silva" email="joao.silva@email.com"
   id="_BvV1BhFfB9W6C8CyXvpAYR" >
      <Role>_xUCzEAilEdyJg53g-AgF8w</Role>
      <Role>_0bj9oAilEdyJg53g-AgF8w</Role>
    </Actor>
    <Actor name="JosÃ© Nunes" email="jose.nunes@email.com"
   id="_sM5jeqXIf8DO3PKZa984Qa" >
      <Role>_2mmR0AilEdyJg53g-AgF8w</Role>
    </Actor>
    <TaskInstance name="Draft User Interface" id="_jDrPUQimEdyJg53g-
   AgF8w,_97at0AimEdyJg53g-AgF8w" idDescriptor="_97at0AimEdyJg53g-AgF8w"
   startDate="28/05/2007" finishDate="08/06/2007" >
        <Task>_MLyRkAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <TaskInstance name="Refine User Interface" id="_jDrPUQimEdyJg53g-
   AgF8w,_DpPMQAinEdyJg53g-AgF8w" idDescriptor="_DpPMQAinEdyJg53g-AgF8w"
   startDate="18/06/2007" finishDate="28/06/2007" >
        <Task>_UWN8YAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <TaskInstance name="Define Tech. Requirements" id="_jDrPUQimEdyJg53g-
   AgF8w,_Fz9-cAinEdyJg53g-AgF8w" idDescriptor="_Fz9-cAinEdyJg53g-AgF8w"
   startDate="23/05/2007" finishDate="28/06/2007" >
        <Task>_RE25UAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <TaskInstance name="Build Application" id="_jDrPUQimEdyJg53g-
   AgF8w,_hJgzQAinEdyJg53g-AgF8w" idDescriptor="_hJgzQAinEdyJg53g-AgF8w"
   startDate="28/06/2007" finishDate="31/07/2007" >
        <Task>_ZzNJgAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <TaskInstance name="Design Process Model" id="_jDrPUQimEdyJg53g-
   AgF8w,_9C8NsAimEdyJg53g-AgF8w" idDescriptor="_9C8NsAimEdyJg53g-AgF8w"
   startDate="15/05/2007" finishDate="28/05/2007" >
        <Task>_HAOysAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <TaskInstance name="Define Requirements" id="_jDrPUQimEdyJg53g-
   AgF8w,_xDFJcAimEdyJg53g-AgF8w" idDescriptor="_xDFJcAimEdyJg53g-AgF8w"
   startDate="01/05/2007" finishDate="28/05/2007" >
        <Task>_Ag70AAimEdyJg53g-AgF8w</Task>
    </TaskInstance>
    <Artifact name="User Requirements" id="_5f9roAilEdyJg53g-AgF8w"
   templatePath = "\\document_server\User Requirements.doc">
        <Tool>Open Office</Tool>
    </Artifact>
    <Artifact name="User Work Processes" id="_7LQIwAilEdyJg53g-AgF8w"
   templatePath = "">
    </Artifact>
    <Artifact name="User Interface {refined}" id="_-xxjgAilEdyJg53g-AgF8w"
   templatePath = "">
        <Tool>Gimp</Tool>
    </Artifact>
    <Artifact name="User Interface {draft}" id="_9eLNoAilEdyJg53g-AgF8w"
   templatePath = "">
        <Tool>Gimp</Tool>
    </Artifact>
  </Instancia>
</uma:MethodLibrary>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo