UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS DE OLIVEIRA ARANTES

Apoio Automatizado à Gerência de Projetos em ODE

Vitória
2006

LUCAS DE OLIVEIRA ARANTES

Apoio Automatizado à Gerência de Projetos em ODE

Monografia apresentada à Universidade Federal do Espírito Santo como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, na área de concentração de Sistemas de Informação, sob orientação do professor Ricardo de Almeida Falbo.

Vitória
2006

LUCAS DE OLIVEIRA ARANTES

Apoio Automatizado à Gerência de Projetos em ODE

Aprovada em 30 de Agosto de 2006.

COMISSÃO EXAMINADORA

Prof. Ricardo de Almeida Falbo, D.Sc.
Orientador

Prof. Vítor Estêvão Silva Souza

Profa. Vanessa Battestin Nunes, M.Sc. (CEFETES)

Agradeço primeiramente ao Senhor Robert Zimmerman (*a.k.a* Bob Dylan), por tocar e dissertar (enquanto eu desenvolvia este trabalho) sobre assuntos complexos que fazem qualquer monografia, dissertação de mestrado ou até tese de doutorado parecerem uma piada. Ao Falbo por ter me dado um apoio **extraordinário** e por ter mantido uma paciência igualmente **extraordinária** durante toda a realização desse trabalho. Aos meus pais e a Sandrinha, que mantiveram a paciência e não me expulsaram de casa quando o estresse bateu. À equipe LabES (Júlio, Fabiano, Rodrigo, Aline, Bruno, Alexandre, Thiago, Vítor, Silvano, Geovando, etc) por terem me agüentado por quase dois anos e me dado um apoio tremendo nas várias fases desse projeto. À Projeta Sistemas de Informação que compreendeu por várias vezes minha necessidade de ficar em casa e revisar mais um documento. À minha atual banda de *rock'n roll*, que me ajudou fortemente a manter a minha sanidade mental me colocando em festas e reuniões com resultados que até Deus duvida. Aos amigos, colegas, parentes, enfim a todos aqueles se importaram com o meu bem-estar e que não paravam de perguntar "pra quando é mesmo?", eu agradeço do fundo do meu coração.

RESUMO

Uma das principais atividades da engenharia de software é o planejamento do projeto, que começa com a definição do escopo e do processo do projeto. A partir dessa definição, é possível derivar várias informações baseadas na decomposição do produto e do processo, tais como: estimativas, alocação de recursos etc. A realização de estimativas, em especial, demanda grande experiência do gerente de projeto e conhecimento sobre projetos passados para uma possível comparação com o projeto atual. Assim, a necessidade de ferramentas que apóiem as diversas atividades do planejamento, provendo dados de projetos anteriores, é de extrema importância.

O objetivo deste trabalho foi desenvolver funcionalidades de apoio ao planejamento para o ambiente de desenvolvimento de software ODE. Inicialmente, foi desenvolvida uma ferramenta de decomposição do produto que permite dividir o projeto em sub-sistemas, módulos e sub-módulos. A partir dessa funcionalidade, evoluiu-se a ferramenta de Estimativa por Análise de Pontos de Função para permitir estimar conjuntos de módulos, escolhendo diferentes formas de contagem. Além disso, evoluiu-se a ferramenta de apoio a Estimativas por Pontos de Casos de Uso, integrando-a, ainda, à ferramenta de modelagem UML do ambiente. Finalmente foi elaborada uma ferramenta que possibilita a elaboração de Estruturas Analíticas de Trabalho (EAT) que tem o intuito de apoiar o gerente de projeto dando uma visão geral sobre a organização do projeto e provendo informações sobre o andamento do mesmo.

SUMÁRIO

ÍNDICE DE FIGURAS

Capítulo 1

Introdução

A Engenharia de Software está se tornando uma atividade cada vez mais complexa e, com sua complexidade, cresceu também o número de métodos para apoiar o desenvolvimento de software. Uma das principais atividades da engenharia de software é o planejamento do projeto, que envolve a definição do processo, isto é, a determinação de um conjunto de atividades para guiar todo o desenvolvimento, sendo que cada uma dessas atividades tem suas tarefas definidas. Além disso, no planejamento inicial o problema é entendido e, normalmente, decomposto em partes menores (decomposição do produto), com o intuito de facilitar o gerenciamento. Com o processo e o escopo do produto de software definidos, é possível tratar várias informações gerenciais tomando por base essas formas de decomposição, tais como estimativas baseadas no processo ou produto, alocação de recursos, acompanhamento das atividades etc.

A realização de estimativas, em especial, demanda grande experiência do gerente de projetos e, além disso, conhecimento sobre projetos passados para uma possível comparação com o projeto atual. Assim, a necessidade de uma ferramenta que apóie de forma automatizada a realização de estimativas e que proveja dados de projetos anteriores é de extrema importância. Contudo, as informações relevantes para essa ferramenta, muitas vezes, estão disponíveis em outras ferramentas. Assim, é importante que elas estejam integradas. Neste contexto, se torna importante que a integração se dê no âmbito de um Ambiente de Desenvolvimento de Software (ADS), já que este visa a integrar várias ferramentas CASE num só ambiente, trazendo inúmeros benefícios à organização. Um exemplo de ambiente que busca atingir as metas de ADSs é o ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2004). A grande diferença entre os ADSs encontrados e ODE é que a fundamentação do último é dada por um conjunto de ontologias.

Este trabalho visa à criação e manutenção de ferramentas do ambiente ODE que apóiem de forma automatizada o gerenciamento de projetos, abrangendo a realização de estimativas, decomposição do produto de software e a criação de Estruturas Analíticas de Trabalho (EATs).

1.1 Motivação

O planejamento inicial de um projeto é uma das fases mais importantes para o sucesso de um projeto de software, já que nela são elaboradas as estimativas de esforço, tempo e custo do mesmo. As estimativas são usadas como insumo para a geração de inúmeros artefatos ainda na fase de planejamento, a saber uma Estrutura Analítica de Trabalho e o cronograma de atividades do projeto. A carga de trabalho sob responsabilidade do gerente de projeto nessa fase é grande e ferramentas de apoio à gerência podem diminuir o peso sobre os ombros desse recurso humano. Além disso, é interessante que as estimativas criadas tenham como foco de contagem partes gerenciáveis do projetos (módulos, por exemplo), fazendo-se necessária a decomposição do produto.

Uma outra forma de apoiar a gerência é permitir uma visualização geral do andamento, esforço despendido, custo atual etc do projeto atual. As Estruturas Analíticas de Trabalho (EATs) têm o intuito de prover essa visão.

1.2 Contexto e Objetivos do Trabalho

Este trabalho foi desenvolvido no contexto do Projeto ODE (FALBO et al., 2004), desenvolvido no Laboratório de Engenharia de Software (LabES) do Departamento de Informática (DI) da Universidade Federal do Espírito Santo (UFES).

No Projeto ODE são integradas várias ferramentas de apoio à gerência de projetos, dentre elas uma ferramenta de Estimativas por Pontos de Função (EPF), inicialmente proposta em (CRUZ, 2001) e uma ferramenta de Estimativas por Pontos de Caso de Uso (EPCU) desenvolvida em (LAHAS, 2005). Essas ferramentas, apesar de muito importantes no ambiente, têm falhas relacionadas à facilidade de uso por possuírem uma interface não intuitiva, conforme relato de organizações parceiras do projeto, e por isso

necessitam de uma re-estruturação para melhor guiar o gerente de projetos na realização de uma estimativa. Além disso, a Ferramenta de Estimativas por Pontos de Função só permite contagens para o projeto de desenvolvimento como um todo e necessitava de uma ferramenta de apoio que permitisse que o produto de software fosse decomposto em módulos, dando ao ambiente a característica de contagem de menores partes do sistema-alvo, facilitando o entendimento da estimativa e a contagem como um todo.

Apesar da variedade de ferramentas de apoio à gerência de projetos, o ambiente ODE necessita de mais integração entre as mesmas e por isso é proposta uma manutenção no ambiente de estimativas (EstimaODE), incluindo também a elaboração de uma ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho (EATs).

Assim, são objetivos deste trabalho:

- Evoluir as ferramentas de apoio à realização de estimativas por pontos de função e pontos de caso de uso, o que inclui o desenvolvimento de uma interface que permita um fluxo mais intuitivo no uso dessas ferramentas;
- O desenvolvimento de uma ferramenta que apóie a Decomposição do Produto de Software e a integração da mesma com a Ferramenta de Estimativa por Ponto de Função;
- O desenvolvimento de uma nova ferramenta de apoio à gerência de projetos que apóie a elaboração de EATs.

1.3 Metodologia

A metodologia de trabalho adotada consistiu, basicamente, de uma revisão bibliográfica, incluindo um estudo sobre o ambiente ODE, desenvolvimento de uma ferramenta de apoio à decomposição do produto, re-estruturação das ferramentas de apoio à realização de estimativas por pontos de função e pontos de caso de uso já existentes, desenvolvimento de uma ferramenta de apoio à elaboração de EATs e a redação da monografia.

Na revisão bibliográfica foram estudados artigos, monografias, dissertações de mestrado, livros e *sites* da Internet que tinham relação com os seguintes assuntos:

Estimativas, Ferramentas CASE, Estrutura Analítica de Trabalho, Ambiente de Desenvolvimento de Sistemas, Planejamento e Gerência de Projeto.

A partir da revisão bibliográfica foi elaborado um plano de integração entre as ferramentas criadas e as re-estruturadas. Conforme citado anteriormente, o escopo definido contaria com o desenvolvimento de ferramentas de apoio à Decomposição do Produto de Software e à elaboração de Estruturas Analíticas de Trabalho e a re-estruturação das ferramentas de Estimativas por Pontos de Função, proposta em (CRUZ, 2001), e Pontos de Caso de Uso, proposta em (LAHAS, 2005). Ainda é levada em conta a integração entre as ferramentas de estimativas re-estruturadas e a integração da Ferramenta de Estimativas por Pontos de Função com a ferramenta de Decomposição do Produto. Todo o processo realizado comportou a realização das atividades de especificação de requisitos, análise, projeto, implementação e testes. As três primeiras atividades permitiram que a implementação e a redação da monografia fossem sendo efetuadas paralelamente.

1.4 Organização do Trabalho

Além deste capítulo, existem seis capítulos e dois anexos neste trabalho.

O Capítulo 2 – *Apoio Automatizado à Gerência de Projetos* – aborda os temas gerência de projetos e automatização do processo de gerência de projetos. Além disso, apresenta informações específicas para este trabalho.

O Capítulo 3 – *O Processo de Software Adotado* – apresenta o processo de software adotado neste trabalho, detalhando ciclo de vida e atividades do processo de software.

O Capítulo 4 – *Especificação de Requisitos* – apresenta a especificação de requisitos funcionais para as duas novas ferramentas contempladas neste trabalho, detalhando modelos de caso de uso e descrições dos mesmos. Além disso, são apresentados os requisitos não-funcionais considerados no trabalho como um todo.

O Capítulo 5 – *Análise* – apresenta os modelos de análise produzidos neste trabalho para as duas novas ferramentas desenvolvidas.

O Capítulo 6 – *Projeto, Implementação e Testes* – apresenta os resultados da fase de projeto considerando aspectos tecnológicos da plataforma de implementação. Além disso, discute-se como foi conduzida a fase de testes.

O Capítulo 7 – *Conclusões e Perspectivas Futuras* – discute algumas considerações finais sobre o projeto e apresenta propostas para trabalhos futuros.

O Anexo A – *Documentação da Evolução da Ferramenta de Apoio a Estimativas Usando Pontos de Função* – apresenta a especificação de requisitos funcionais, análise, projeto e implementação para a evolução da ferramenta de Estimativa de Pontos de Função.

O Anexo B – *Documentação da Evolução da Ferramenta de Apoio a Estimativas Usando Pontos de Caso de Uso* – apresenta a especificação de requisitos funcionais, análise, projeto e implementação para a evolução da ferramenta de Estimativa de Pontos de Caso de Uso.

Capítulo 2

Apoio Automatizado à Gerência de Projetos

No planejamento de um projeto, gerentes de projeto e analistas trabalham de forma unificada na elaboração de um projeto de qualidade dentro dos limites estabelecidos pelo(s) cliente(s). Nesse momento as atenções estão normalmente voltadas ao entendimento do problema e à elaboração de um arcabouço de informações que permita esclarecer diversas informações para a organização desenvolvedora e o(s) cliente(s). O gerente de projeto fica normalmente em foco nessa fase, pois é parte do seu trabalho a elaboração de estimativas de esforço, custo e tempo, alocação de recursos e montagem de equipes, análise dos riscos do projeto, confecção de cronogramas etc. Erros nessa fase podem representar grande risco para o sucesso do projeto e, portanto, se faz necessário o uso de ferramentas que apóiem a gerência de projetos, automatizando algumas de suas tarefas e evitando erros.

Este capítulo é responsável por mostrar as várias áreas estudadas e contempladas neste trabalho. A organização do mesmo é dada da seguinte forma: a seção 2.1 aborda a Gerência de Projetos e as áreas de conhecimento do Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos – O Guia PMBOK (*Project Management Body Of Knowledge*) (PMI, 2004); na seção 2.2 a discussão sobre a área de conhecimento de gerência de escopo é aprofundada; a seção 2.3 disserta sobre as estimativas como um todo e os métodos contemplados neste trabalho; na seção 2.4 as Estruturas Analíticas de Trabalho (EATs) são abordadas; a seção 2.5 abre uma discussão sobre a Automatização do Processo; a seção 2.6 descreve o ambiente ODE e suas ferramentas e, finalmente, a seção 2.7 discute sobre a Automatização da Gerência de Projetos em ODE.

2.1 Gerência de Projetos

Gerência de Projetos é a aplicação de conhecimentos e técnicas organizadas de forma a alcançar objetivos pré-estabelecidos. Inicialmente usada em projetos bélicos e

aeroespaciais pelo Departamento de Defesa Americano (desde 1960), os bons resultados obtidos nessa área fizeram com que a gestão de projetos se expandisse e, finalmente ganhasse atenção significativa em grande parte (senão todas) das áreas tecnológicas (MARTINS, 2005). Um outro motivo do ganho de atenção dado a essa disciplina é o fato de que as organizações tendem, com a globalização tecnológica e comercial, a ficar mais competitivas e, por isso, visam obter melhores resultados (tempo e custo minimizados e qualidade aumentada) nos projetos executados.

Devido ao crescimento do interesse nessa disciplina foi inevitável o surgimento de uma entidade que procurasse regular os conhecimentos relacionados a essa área. O PMI (*Project Management Institute*) é um órgão internacional que visa promover o profissionalismo e desenvolver o "estado-da-arte" na gestão de projetos (MARTINS, 2005), especificando um conjunto de procedimentos de forma a padronizar a teoria do gerenciamento de projetos. Foi elaborado, então, um corpo de conhecimento sobre gerenciamento de projetos, o PMBOK (*Project Management Body of Knowledge*), que tem como objetivo principal identificar e agrupar os conhecimentos sobre a profissão, além de padronizar o vocabulário da mesma. No PMBOK, foram especificadas nove áreas de conhecimento relacionadas a essa disciplina (PMI, 2004):

1. **Gerência de Integração**: a área de conhecimento em gerenciamento de integração do projeto inclui os processos e as atividades necessárias para identificar, definir, combinar, unificar e coordenar os diversos processos e atividades de gerenciamento de projetos dentro dos grupos de processos de gerenciamento de projetos. Neste contexto, a integração inclui características de unificação, consolidação, articulação e ações integradoras que são essenciais para o término do projeto, para atender com sucesso às necessidades do cliente e de outras partes interessadas e para gerenciar as expectativas. A integração, no contexto do gerenciamento de um projeto, consiste em fazer escolhas sobre em que pontos concentrar recursos e esforço, antecipando possíveis problemas, tratando-os antes de se tornarem críticos e coordenando o trabalho, visando o bem geral do projeto. O esforço de integração também envolve fazer compensações entre objetivos e alternativas conflitantes;
2. **Gerência de Escopo:** o gerenciamento do escopo do projeto inclui os processos necessários para garantir que o projeto inclua todo o trabalho necessário, e somente

ele, para terminar o projeto com sucesso. O gerenciamento do escopo do projeto trata principalmente da definição e controle do que está e do que não está incluído no projeto;

3. **Gerência de Tempo:** o gerenciamento de tempo do projeto inclui os processos necessários para realizar o término do projeto no prazo. Os processos contemplados nessa área de conhecimento são:
    - **Definição de atividades:** identificação das atividades específicas do cronograma que precisam ser realizadas para produzir as várias entregas do projeto;
    - **Seqüenciamento de atividades**: identificação e documentação das dependências entre as atividades do cronograma;
    - **Estimativa de recursos das atividades:** estimativa do tipo e das quantidades de recursos necessários para realizar cada atividade do cronograma;
    - **Estimativa de duração das atividades:** estimativa do número de períodos de trabalho que serão necessários para terminar as atividades individuais do cronograma;
    - **Desenvolvimento do cronograma**: análise dos recursos necessários, restrições do cronograma, durações e seqüências de atividades para criar o cronograma do projeto;
    - **Controle do cronograma:** controle das mudanças no cronograma do projeto.
4. **Gerência de Custos:** o gerenciamento de custos do projeto inclui os processos envolvidos no planejamento, estimativa, orçamentação e controle de custos, de modo que seja possível terminar o projeto dentro do orçamento aprovado. Os seguintes processos são contemplados nessa área de conhecimento:
    - **Estimativa de custos:** desenvolvimento de uma estimativa dos custos necessários para terminar as atividades do projeto;
    - **Orçamentação:** agregação dos custos estimados de atividades individuais ou pacotes de trabalho para estabelecer uma linha de base dos custos;

- **Controle de custos:** controle dos fatores que criam as variações de custos e controle das mudanças no orçamento do projeto.

5. **Gerência da Qualidade:** os processos de gerenciamento da qualidade do projeto incluem todas as atividades da organização executora que determinam as responsabilidades, os objetivos e as políticas de qualidade, de modo que o projeto atenda às necessidades que motivaram sua realização. Eles implementam o sistema de gerenciamento da qualidade através da política, dos procedimentos e dos processos de planejamento da qualidade, garantia da qualidade e controle da qualidade, com atividades de melhoria contínua dos processos conduzidas do início ao fim, conforme adequado.
6. **Gerência de Recursos Humanos:** o gerenciamento de recursos humanos do projeto inclui os processos que organizam e gerenciam a equipe do projeto. Os processos contemplados nessa área de conhecimento são:
    - **Planejamento de recursos humanos:** identificação e documentação de funções, responsabilidades e relações hierárquicas do projeto, além da criação do plano de gerenciamento de pessoal;
    - **Contratar ou mobilizar a equipe do projeto:** obtenção dos recursos humanos necessários para terminar o projeto;
    - **Desenvolver a equipe do projeto:** melhoria de competências e interação de membros da equipe para aprimorar o desempenho do projeto;
    - **Gerenciar a equipe do projeto:** acompanhamento do desempenho de membros da equipe, fornecimento de feedback, resolução de problemas e coordenação de mudanças para melhorar o desempenho do projeto.
7. **Gerência de Comunicação:** o gerenciamento das comunicações do projeto é a área de conhecimento que emprega os processos necessários para garantir a geração, coleta, distribuição, armazenamento, recuperação e destinação final das informações sobre o projeto de forma oportuna e adequada.
8. **Gerência de Riscos:** o gerenciamento de riscos do projeto inclui os processos que tratam da realização de identificação, análise, respostas, monitoramento e controle e planejamento do gerenciamento de riscos em um projeto. A maioria desses processos é atualizada durante todo o projeto. Os objetivos do gerenciamento de

riscos do projeto são aumentar a probabilidade e o impacto dos eventos positivos e diminuir a probabilidade e o impacto dos eventos adversos ao projeto.

9. **Gerência de Aquisições:** o gerenciamento de aquisições do projeto inclui os processos para comprar ou adquirir os produtos, serviços ou resultados necessários de fora da equipe do projeto. Além disso, são contemplados os processos de gerenciamento de contratos e de controle de mudanças necessários para administrar os contratos ou pedidos de compra emitidos por membros autorizados da equipe do projeto. O gerenciamento de aquisições do projeto também inclui a administração de qualquer contrato emitido por uma organização externa (o comprador) que está adquirindo o projeto da organização executora (o fornecedor) e a administração de obrigações contratuais estabelecidas para a equipe do projeto pelo contrato.

Neste trabalho focaremos nas disciplinas de gerência de escopo, gerência de tempo e gerência da integração.

2.2 Gerência de Escopo

Segundo o PMBOK (PMI, 2004), o termo escopo tem um significado especial para o foco sobre o produto e o foco sobre o projeto. No contexto de produto, o escopo é tratado como as características e funções que descrevem o produto, serviço ou resultado. No contexto de projeto, o escopo se refere ao trabalho que precisa ser realizado para entregar um produto, serviço ou resultado com as características e funções especificadas. Seguindo o contexto de projeto, a área de conhecimento de Gerência de Escopo contempla os processos com suas ferramentas e técnicas que gerenciam o escopo do projeto como um todo, garantindo definição, controle e manutenção do mesmo. Os seguintes processos são encontrados nessa área de conhecimento (PMI, 2004):

- **Planejamento do Escopo**: criação de um plano de gerenciamento do escopo do projeto que trata de como o escopo do projeto será definido, verificado e controlado e como a estrutura analítica de trabalho (EAT) será criada e definida;
- **Definição do Escopo:** desenvolvimento de uma declaração detalhada do escopo do projeto para servir de base para futuras decisões do projeto;

- **Criação da EAT:** subdivisão das principais entregas do projeto e do trabalho do projeto em componentes menores e mais facilmente gerenciáveis;
- **Verificação do Escopo:** formalização da aceitação das entregas do projeto concluídas;
- **Controle do Escopo:** controle das mudanças no escopo do projeto.

Todos os processos citados interagem entre si e com processos de outras áreas de conhecimento.

2.3 Estimativas

A atividade primária e primordial do desenvolvimento de qualquer projeto seja ele de software ou não, é o planejamento do mesmo. Nessa fase devem ser levantadas características do projeto cruciais para o bom andamento do mesmo. As estimativas são algumas das tarefas dessa atividade e têm por objetivo medir objetos pré-definidos (tempo, custo, esforço, recursos etc).

A elaboração de uma estimativa pode ser influenciada por vários fatores como: alterações nos requisitos, mudanças na equipe de desenvolvimento, domínio da tecnologia empregada etc. Só é possível se ter exatidão em uma estimativa no fim do projeto e se deve garantir que as estimativas são aprimoradas durante todo o ciclo de vida do projeto. Algumas opções para a melhoria na busca de boas estimativas são (CARVALHO et. al, 2006): (i) Usar técnicas de decomposição; (ii) Usar um ou mais métodos de estimativas; (iii) Basear as estimativas em projetos similares que já tenham sido concluídos.

Técnicas de decomposição dividem o problema em partes menores na realização de estimativas, decompondo um projeto em suas funções (decomposição do produto) ou atividades (decomposição do processo) principais. Decompondo o escopo do projeto, o gerente de projeto tem unidades menores a estimar, o que é mais fácil de estimar do que o projeto como um todo.

Métodos de estimativa definem uma abordagem sistemática para se estimar alguma grandeza relativa ao projeto de software. Há diversos métodos de estimativas propostos na literatura, sendo que aqueles focados em estimativas de tamanho têm sido

bastante utilizados. A estimativa de tamanho é uma das estimativas mais utilizadas pela sua versatilidade, podendo ser transformada em esforço, tempo e custo. Dentre os métodos de estimativas de tamanho destacam-se a Análise de Pontos de Função e a Análise de Pontos de Casos de Uso.

Uma estimativa de tamanho não é uma medida diretamente palpável, mas é possível usá-la como insumo para a elaboração de estimativas de esforço (homens-hora, homens-dia etc) aplicando sobre a estimativa alcançada um fator de produtividade, dependente do método de estimativa de tamanho usado (Pontos de Função, Pontos de Caso de Uso etc). Com a estimativa de esforço em mãos, pode-se obter estimativas de tempo e custo.

Neste trabalho abordaremos unicamente as estimativas de tamanho, com enfoque nos métodos de Análise de Pontos de Função e Análise de Pontos de Caso de Uso.

2.3.1 - Análise de Pontos de Função

A Análise de Pontos de Função (APF) é um método surgido no início da década de 1970 no parque de pesquisas da IBM. Foi inicialmente usada para medir a produtividade da equipe num ambiente que havia grande número de projetos e grande número de linguagens de programação usadas (VAZQUEZ et. al, 2005). Posteriormente foi criado um órgão internacional, o IFPUG – *International Function Point Users Group,* com o intuito de pesquisar o método e padronizá-lo, devido à crescente necessidade das organizações estimarem esforço, tempo e custos de seus projetos.

O método, como proposto pelo IFPUG, tem como base o seguinte conjunto de atividades, cuja ordem e dependências são mostradas na Figura 1 (FALBO, 2005):

- **Determinar o tipo de contagem de pontos de função**: este é o primeiro passo no processo de contagem, sendo que existem três tipos de contagem: contagem de pontos de função (PFs) de projeto de desenvolvimento, de aplicações instaladas e de projetos de manutenção;

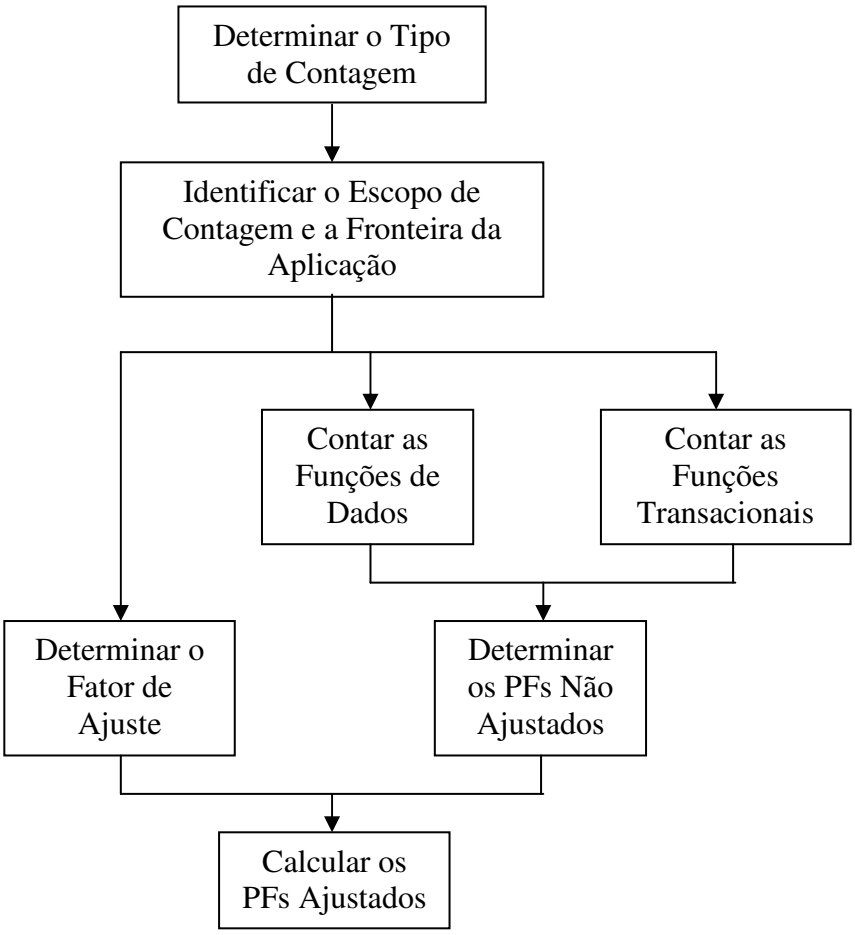

Figura 2.1 – O Processo de Contagem de Pontos de Função segundo o IFPUG (FALBO, 2005).

- **Identificar o escopo de contagem e a fronteira da aplicação**: neste passo, definem-se as funcionalidades que serão incluídas em uma contagem de PFs específica. A fronteira da aplicação é definida estabelecendo um limite lógico entre a aplicação que está sendo medida, os usuários e outras aplicações. O escopo de contagem define a parte do sistema (funcionalidades) a ser contada;
- **Determinar a contagem de pontos de função não ajustados**: os pontos de função não ajustados refletem as funcionalidades fornecidas pelo sistema para o usuário. Essa contagem leva em conta dois tipos de função: de dados e transacionais, bem como sua complexidade (simples, média ou complexa);

- **Contar as funções de dados**: as funções de dados representam as funcionalidades relativas aos requisitos de dados internos e externos à aplicação. São elas os Arquivos Lógicos Internos (ALIs) e os Arquivos de Interface Externa (AIEs). Ambos são grupos de dados logicamente relacionados ou informações de controle que foram identificados pelo usuário. A diferença está no fato de um ALI ser mantido dentro da fronteira da aplicação, isto é, armazenar os dados mantidos através de um ou mais processos elementares da aplicação, enquanto que um AIE é apenas referenciado pela aplicação, ou seja, ele é mantido dentro da fronteira de outra aplicação. Assim, o objetivo de um AIE é armazenar os dados referenciados por um ou mais processos elementares da aplicação sendo contada, mas que são mantidos por outras aplicações. A contagem das funções de dados envolve a identificação de ALIs e AIEs e a classificação de cada uma dessas funções de dados segundo sua complexidade funcional. Para maiores detalhes sobre essa classificação, vide (VAZQUEZ et al., 2005);
- **Contar as funções transacionais**: as funções transacionais representam as funcionalidades de processamento de dados do sistema fornecidas para o usuário. São elas: as Entradas Externas (EEs), as Saídas Externas (SEs) e as Consultas Externas (CEs). As EEs são processos elementares que processam dados (ou informações de controle) que entram pela fronteira da aplicação. O objetivo principal de uma EE é manter um ou mais ALIs ou alterar o comportamento do sistema. As SEs são processos elementares que enviam dados (ou informações de controle) para fora da fronteira da aplicação. Seu objetivo é mostrar informações recuperadas através de um processamento lógico (isto é, que envolva cálculos ou criação de dados derivados) e não apenas uma simples recuperação de dados. Uma SE pode, também, manter um ALI ou alterar o comportamento do sistema. Por fim, uma CE, assim como uma SE, é um processo elementar que envia dados (ou informações de controle) para fora da fronteira da aplicação, mas sem realização de nenhum cálculo nem a criação de dados derivados. Seu objetivo é apresentar informação para o usuário, por meio apenas de uma recuperação das informações. Nenhum ALI é mantido durante sua realização, nem o comportamento do sistema alterado. A contagem das funções transacionais envolve a identificação de EEs, SEs

e CEs e a classificação de cada uma dessas funções segundo sua complexidade funcional. Para maiores detalhes sobre essa classificação, vide (VAZQUEZ et al., 2005);

- **Determinar o valor do fator de ajuste**: o fator de ajuste é baseado em 14 características gerais de sistemas, que avaliam a funcionalidade geral da aplicação que está sendo contada, e seus níveis de influência. O nível de influência de uma característica é determinado com base em uma escala de 0 (nenhuma influência) a 5 (forte influência). Assim, o fator de ajuste visa a ajustar os pontos de função não ajustados em ± 35%. Esse passo tornou-se opcional em 2002 para que o método da Análise de Pontos de Função passasse a ser um padrão internacional de medição funcional (ISO/IEC 20926). As principais críticas são a grande variação na interpretação das 14 características gerais de sistemas e a constatação que algumas delas estão desatualizadas;
- **Calcular os pontos de função ajustados**: finalmente, os PFs ajustados são calculados, considerando-se o tipo de contagem definido no primeiro passo e o fator de ajuste.

Um dos maiores problemas associados à Análise de Pontos de Função é que os dados necessários para essa análise são bastante imprecisos no início de um projeto. Assim, pode ser pouco produtivo utilizar o método integralmente para realizar as primeiras estimativas. De fato, a Análise de Pontos de Função é, antes de mais nada, um método de medição e, portanto, seu uso para a realização de estimativas de tamanho é uma adaptação (FALBO, 2005). Tendo em vista isso, foram propostas algumas variações mais simples do método voltadas para a realização de estimativas e que apresentam resultados satisfatórios, dentre elas a Contagem Estimativa da NESMA (*NetherLans Software Metrics Users Association*) (VAZQUEZ et al., 2005).

A abordagem da NESMA, ao contrário da abordagem segundo o IFPUG, não leva em conta qualquer característica especial do projeto, necessitando apenas de uma classificação dos objetos obtidos através de uma pesquisa no escopo do projeto para a execução de uma função e geração da quantidade de pontos de função, agilizando o processo de estimativa de tamanho. Um possível processo para a Contagem Estimativa da NESMA é mostrado na Figura 2.2.

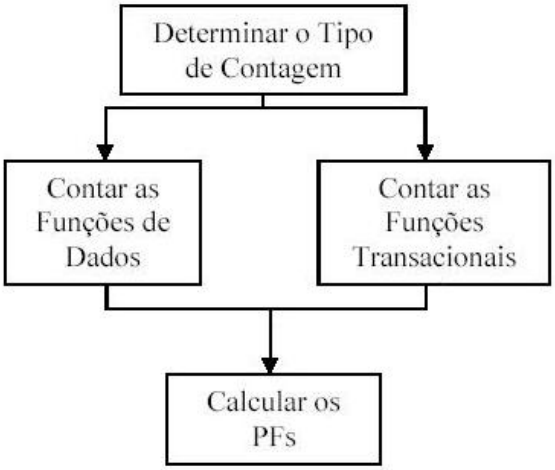

Figura 2.2 - Processo da Contagem Estimativa da NESMA.

Vale destacar que os passos que envolvem a contagem das funções transacionais e de dados não requerem uma classificação das mesmas em termos de sua complexidade funcional, sendo necessário, portanto, apenas identificar ALIs, AIES, EEs, SEs e CEs. O cálculo dos pontos de função segue a seguinte fórmula para obtenção dos mesmos:

$$numPF = 7*numALI + 5*numAIE + 4*numEE + 5*numSE + 4*numCE$$

onde *numPF* é o total de pontos de função não ajustados calculado e *numALI*, *numAIE*, *numEE*, *numSE* e *numCE* correspondem, respectivamente, aos números de ALIs, AIEs, EEs, SEs e CEs do projeto em questão.

2.3.2 - Análise de Pontos de Casos de Uso

A Análise dos Pontos de Caso de Uso (APCU) é um método de estimativa de tamanho de projetos de software orientados a objetos, baseada nas técnicas de Análise de Pontos de Função e Modelagem de Casos de Uso (ANDRADE, 2004). O método explora o modelo e a descrição dos casos de uso, retirando informações necessárias para a confecção da estimativa. Basicamente, a APCU envolve os seguintes passos, mostrados na Figura 2.3 (SCHNEIDER et al., 2001):

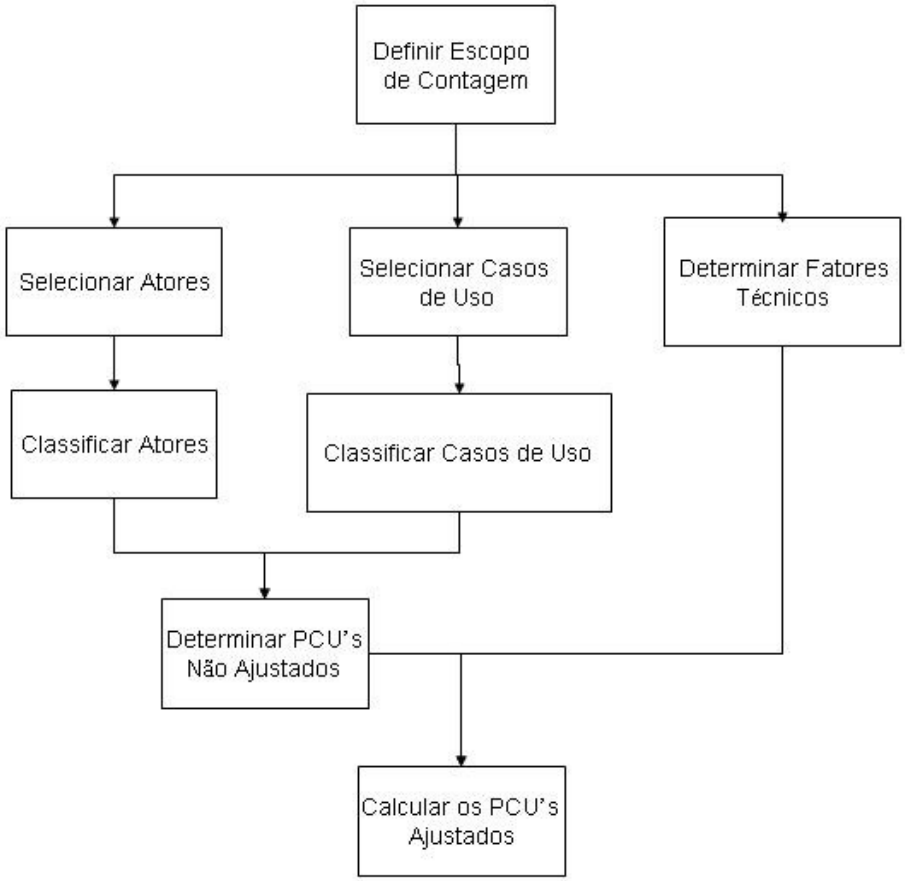

Figura 1.3 – Processo de Contagem de Estimativa de Pontos de Caso de Uso

1 **Determinar o peso dos atores**: inicialmente, cada ator envolvido no sistema é classificado em uma das seguintes categorias[1]: simples (1), médio (2) ou complexo (3). Em seguida contam-se quantos atores há de cada tipo (qteAtor) e multiplica-se esse valor pelo correspondente fator de peso de ator (fpAtor), obtendo-se o peso dos atores (PA).

$$PA = \sum_{i=1}^{3} fpAtor_i * qteAtor_i$$

[1] Os números entre parênteses correspondem ao fator de peso de ator (fpAtor) sugerido para cada categoria em (SCHNEIDER et al., 2001).

2 **Determinar o peso dos casos de uso**: determina-se a complexidade[2] de cada caso de uso – simples (5), médio (10) e complexo (15) – tomando por base o número de transações do caso de uso, incluindo cenários secundários, ou o número de classes de análise usadas para implementar o caso de uso. A seguir, contam-se quantos casos de uso há de cada tipo (qteCasoUso) e multiplica-se esse valor por seu fator de peso de caso de uso (fpCasoUso), obtendo-se o peso dos casos de uso (PT).

$$PT = \sum_{i=1}^{3} fpCasoUso_i * qteCasoUso_i$$

3 **Determinar a contagem de pontos de caso de uso não ajustados**: somando-se PA e PT, tem-se os pontos de caso de uso não ajustados (PCUNA).

$$PCUNA = PA + PT$$

4 **Determinar fatores técnicos**: envolve o cálculo dos pesos de fatores técnicos, a saber Fator Técnico de Complexidade (FTC) e Fator Ambiental (FA). O primeiro, como o nome indica, trata da complexidade técnica do projeto e envolve fatores como distribuição, desempenho, reutilização, portabilidade etc. O segundo refere-se à experiência das pessoas envolvidas, incluindo fatores como conhecimento e experiência na linguagem de programação, dedicação etc. Para maiores detalhes sobre o cálculo desses fatores, vide (SCHNEIDER et al., 2001).

5 **Calcular os pontos de caso de uso ajustados**: finalmente são calculados os pontos de caso de uso ajustados (PCU) do projeto, por meio da fórmula abaixo:

$$PCU = PCUNA * FTC * FA$$

[2] Os números entre parênteses correspondem ao fator de peso de caso de uso (fpCasoUso) sugerido para cada categoria em (SCHNEIDER et al., 2001).

2.4 Estrutura Analítica de Trabalho

Uma Estrutura Analítica de Trabalho (EAT), em inglês, *Work Breakdown Structure* – WBS, é uma ferramenta de gerenciamento que contém uma série de informações que varia no nível de detalhamento desejado em termos do escopo do projeto em questão. Algumas das vantagens do uso do EAT são (MARTINS, 2005):

- Pode representar os produtos finais e subprodutos que serão entregues;
- Permite uma visualização detalhada do escopo do projeto;
- Pode ser usada na elaboração do cronograma e permite monitorar o progresso do projeto;
- Pode ser usada no detalhamento de custos de cada tarefa;
- Apóia o gerente na elaboração e distribuição das equipes de trabalho;
- Facilita a identificação de riscos durante o projeto.

A construção de uma EAT se inicia pela decomposição do projeto em grupos de subprojetos recursivamente até que se obtenha o nível de detalhe desejado. É interessante que essa decomposição divida tarefas complexas em tarefas menores, conhecidas como pacotes de trabalho, que podem ser implementados em um tempo gerenciável de no máximo uma semana. Os níveis de decomposição que contém os pacotes de trabalho são conhecidos como tarefas-resumo (MARTINS, 2005). A Figura 2.3 mostra um exemplo de EAT.

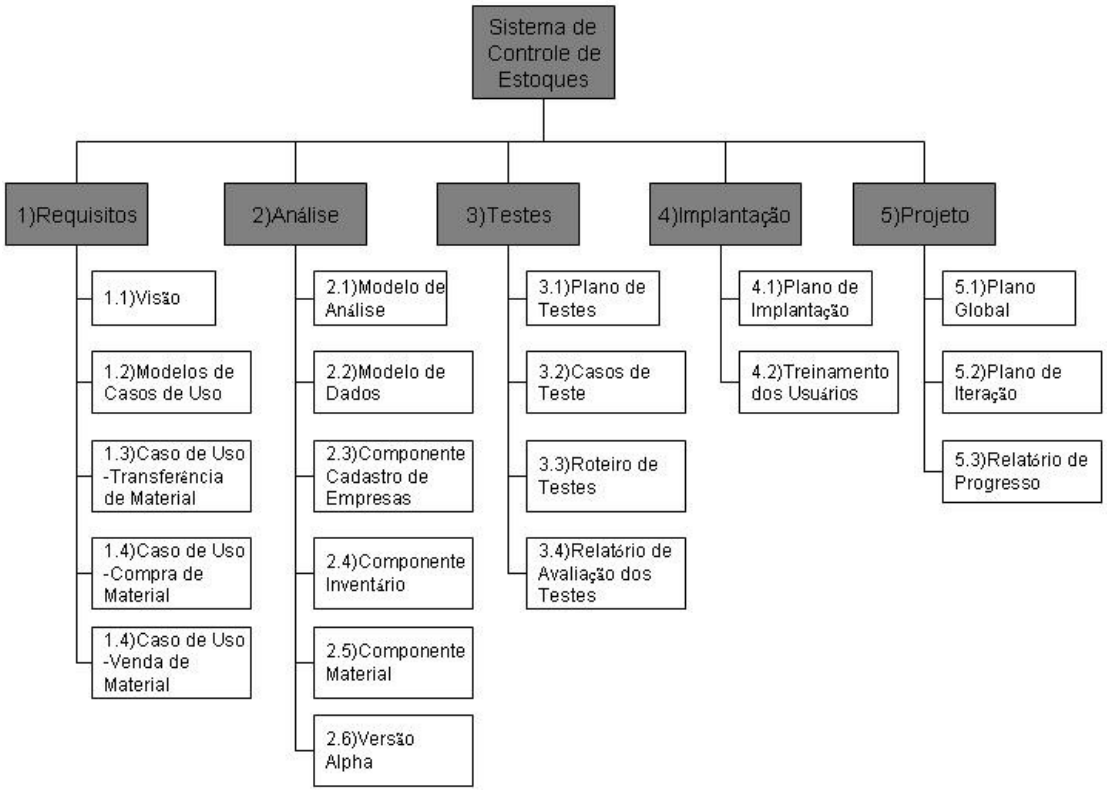

Figura 2.2 - Exemplo de EAT (MARTINS, 2005).

Com o nível de detalhamento alcançado, pode-se preencher os pacotes de trabalho com informações valiosas trazidas de estimativas (tempo, custo, esforço etc), pesquisas com projetos anteriores (riscos possíveis, por exemplo) e andamento atual de cada tarefa. A EAT deve ser atualizada a cada nova informação durante todo o ciclo de vida do desenvolvimento. Posteriormente é possível confrontar as atividades extraídas da EAT com as atividades do processo de software (planejamento, análise, projeto, implementação, testes e implantação). Dessa forma, temos uma matriz informacional de onde se podem extrair informações do tipo: (1) Qual o custo da fase de análise de um dado módulo do produto final? (2) Qual o tempo despendido no planejamento de um dado módulo? (3) Como está o progresso da fase de implementação de cada módulo?

2.5 Automatização do Processo

Nos primórdios da engenharia de software, os responsáveis pelo desenvolvimento de um projeto guardavam todas as informações relacionadas ao andamento do mesmo em planilhas e relatórios escritos em papel, dando margem à ocorrência de muitos erros. Essa forma não automatizada tornou-se inviável à medida que os projetos ganharam tamanho e complexidade, pois a tarefa de manter documentos e planilhas atualizadas passou a tomar um tempo considerável da jornada de trabalho do desenvolvedor. Nesse contexto, surgiram as ferramentas CASE (*Computer Aided Software Engineering*) que têm por objetivo fornecer ao engenheiro de software apoio automatizado às atividades manuais e aperfeiçoar o conhecimento de engenharia (PRESSMAN, 2002). Além disso, o uso de ferramentas CASE pode melhorar a qualidade do produto final, já que tende a evitar possíveis falhas na confecção e atualização de artefatos produzidos ao longo do ciclo de desenvolvimento.

As ferramentas CASE, geralmente, são bastante especializadas e pontuais, apoiando o engenheiro de software em uma atividade específica do processo de software. Com a grande variedade e especialização das ferramentas CASE, é interessante que se tenha uma integração entre ferramentas com o intuito de poupar trabalho e tempo do desenvolvedor no momento de preencher os dados de entrada de uma ferramenta com os dados de saída de outra. Assim, é importante que se tenha um arcabouço de software montado para servir de base para a integração de ferramentas, de modo a permitir o fluxo de dados entre elas, dando agilidade e qualidade ao processo de desenvolvimento.

Neste contexto surgiram os Ambientes de Desenvolvimento de Software (ADSs) que têm o intuito de prover essa infra-estrutura, permitindo, dentre outros (PRESSMAN, 2002): (i) transferência automática de informação entre as ferramentas; (ii) redução no trabalho realizado em atividades de apoio, como garantia da qualidade e gerência de configuração; (iii) melhoria no acompanhamento do projeto.

Para que uma organização tenha seus processos automatizados, é interessante que se tenha um ADS configurável, capaz de se adequar à forma de trabalho e aos processos dessa organização com o intuito de obter o máximo de vantagens no uso do mesmo.

2.6 O Ambiente ODE

ODE (*Ontology-based software Development Environment*) (BERTOLLO, 2002) é um ambiente de desenvolvimento de software centrado em processo, que tem sua fundamentação em um conjunto de ontologias relacionadas do domínio de Engenharia de Software. Esse ambiente vem sendo desenvolvido no Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo (UFES), usando apenas ferramentas livres como a linguagem de programação Java, o sistema de gerenciamento de banco de dados PostgreSQL e o sistema operacional Linux.

A premissa do projeto ODE é a seguinte: se as ferramentas de um ADS são construídas baseadas em ontologias, a integração das mesmas é facilitada, pois os conceitos envolvidos estão formalmente definidos nas ontologias. Essa é uma das principais características que distingue ODE de outros ADSs: sua base ontológica. Especialmente em ADSs, ontologias reduzem confusões terminológicas e conceituais, facilitando o entendimento compartilhado e a comunicação entre pessoas com diferentes necessidades e pontos de vista. Além disso, a padronização de conceitos provida por uma ontologia permite que a comunicação entre as ferramentas que compõem o ambiente seja aprimorada (FALBO et al., 2005).

ODE possui uma infra-estrutura que permite controlar projetos de software e definir seus respectivos processos. Integradas a essa infra-estrutura, existem diversas ferramentas de apoio à construção, gerência e avaliação de qualidade, que apóiam diversas atividades do processo de software. Entre elas podem-se citar as seguintes modalidades de ferramentas: estimativas, análise de riscos, modelagem de análise e projeto, documentação, acompanhamento de processo, gerenciamento de recursos, planejamento e controle da qualidade.

2.7 Apoio Automatizado à Gerência de Projetos em ODE

Para que as tarefas do planejamento de um projeto sejam realizadas de forma correta e acurada, é importante que a organização conte com gerentes experientes que saibam lidar com a execução das mesmas de forma eficaz dentro de um ambiente com todas as variáveis que um projeto pode envolver, tais como prazo diminuído e custo

minimizado. Além do planejamento, o gerente de projetos deve se preocupar com o acompanhamento do projeto, no qual são avaliados e atualizados diversos artefatos gerados e dados provenientes da fase de planejamento.

De fato, contar com profissionais de gerência experientes pode permitir uma melhor administração dos projetos, mas nem sempre isso é possível, pois profissionais com essa característica são escassos no mercado e normalmente têm um custo de hora elevado. A Automatização da Gerência de Projetos tem o objetivo de auxiliar os gerente de projetos, experientes ou não, em suas tarefas, diminuindo sua carga de trabalho e agilizando o processo contínuo de gerência.

O ambiente ODE conta com várias ferramentas para automatizar o processo de gerência de projetos, dentre elas:

- Ferramenta de Apoio à Definição de Processos de Software (BERTOLLO et al., 2006): permite a definição de processos padrão da organização e sua instanciação em cada projeto;
- EstimaODE: apóia a realização de estimativas, contando com ferramentas de apoio à Análise de Pontos de Função (contemplando as contagens elaboradas pelo IFPUG) (CRUZ, 2001) e Análise de Pontos de Caso de Uso (LAHAS, 2005);
- GeRIS: ferramenta de apoio ao gerenciamento de riscos que contempla a identificação, avaliação, planejamento e monitoramento de riscos em um projeto (FALBO et al., 2004);
- ControlPRO: ferramenta de apoio ao acompanhamento de projetos que permite ao gerente vislumbrar o estado de cada atividade do processo, além de realizar tarefas como alocações e alteração dinâmica do processo (DAL MORO et al., 2005);
- ReqODE: ferramenta de apoio ao controle de requisitos de um projeto, contemplando cadastro e gerência de requisitos (MARTINS et al., 2006).

Mesmo com um grande número de ferramentas gerenciais integradas ao ambiente, é interessante que essas sejam melhoradas de forma contínua e, por isso, foram avaliadas algumas oportunidades de melhoria em ODE, chegando-se à conclusão de que a ferramenta EstimaODE podia evoluir.

Antes da realização deste trabalho o processo de contagem da ferramenta de Análise de Pontos de Função, parte integrante de EstimaODE, contemplava somente a abordagem padrão IFPUG. Apesar desse método de contagem ser um dos mais usados na atualidade, os dados necessários para essa contagem são bastante imprecisos no início de um projeto e, portanto, gerentes de projeto são, muitas vezes, obrigados a produzir estimativas antes de um estudo mais aprofundado. Nesse sentido foi estudada a possibilidade da incorporação da Contagem Estimativa da NESMA, que funciona similarmente ao IFPUG, mas tende a acelerar o processo de contagem. Conforme, discutido anteriormente, para realizar uma contagem segundo a abordagem da NESMA, basta identificar as funções de dados e transacionais do sistema e em seguida calcular o número de Pontos de Função usando fatores pré-definidos.

Outra evolução diz respeito à definição do escopo de um projeto. ODE conta com um aparato bastante maduro para definição e acompanhamento de processos. Por outro lado, se o foco não for o processo e sim o produto, com todas suas divisões (sub-sistemas, módulos e sub-módulos, recursivamente), ODE tem uma grande lacuna a ser preenchida e não conta com nenhuma ferramenta para apoiar a decomposição do produto.

São vários os ganhos que o ambiente teria com a incorporação de um maquinário para decomposição do produto, dentre eles: (i) melhor entendimento das funcionalidades a serem desenvolvidas em um projeto; (2) maior entendimento e facilidade na confecção de estimativas, permitindo integrar esse tipo de decomposição com estimativas de Pontos de Função e dados históricos, por exemplo; (3) possibilidade de construção de uma EAT, entendendo as menores partes da decomposição do produto como pacotes de trabalho.

Por fim, apesar do grande número de ferramentas, os usuários de ODE normalmente têm dificuldade de utilizar algumas ferramentas, pois as interfaces implementadas não são, em alguns casos, intuitivas o bastante e normalmente não apóiam o usuário a realizar os passos que devem ser dados para cumprir uma tarefa. Assim, as ferramentas em manutenção serão revisadas, também, sob um aspecto de interface gráfica.

Capítulo 3

O Processo de Software Adotado

À medida que a complexidade e o tamanho de um projeto aumentam ou a equipe de desenvolvimento cresce, a necessidade de se ter organização e metodologia de trabalho é fundamental para que o desenvolvimento de um projeto seja executado com qualidade. Neste contexto surge uma base técnica (métodos, normas, atividades etc) que auxilia a equipe desenvolvedora na produção do software e que visa à melhoria organizacional e à padronização do desenvolvimento. Trata-se do processo de software.

Neste trabalho foi utilizado um processo focado no paradigma orientado a objetos e neste capítulo apresentamos esse processo.

3.1 Processo de Software

O processo de software é o conjunto de atividades, métodos e práticas adotadas para a realização de um projeto de software. Em um processo de software são definidos: (i) as atividades que compõem o processo e a forma com que serão organizadas, tomando por base um modelo de ciclo de vida; (ii) os artefatos gerados e consumidos nas atividades definidas; (iii) os procedimentos adotados para a realização de cada atividade; e (iv) os recursos (humano, software e hardware) para a realização das atividades (FALBO, 1998).

Diante desse conjunto de requisitos em um processo de software, são estabelecidas algumas diretivas, que devem ser seguidas na escolha das atividades. É necessário que o processo escolhido contemple, minimamente as seguintes atividades (PRESSMAN, 2002):

- Planejamento: é o ponto de partida da linha de desenvolvimento de um projeto de software, além de estar presente durante todo o ciclo de vida de um projeto na forma de um acompanhamento. Nessa fase são definidas as atividades que farão parte do processo, o modelo de ciclo de vida para a

realização das mesmas, a definição dos recursos necessários, a confecção de estimativas, o gerenciamento dos riscos e a elaboração de um cronograma;

- Levantamento de requisitos: é responsável pela captura e entendimento dos requisitos expostos pelos clientes, além da definição de um escopo do software-alvo. Nessa fase são levados em conta, também, requisitos não funcionais, tais como desempenho do produto final, restrições, tipo de interface requisitada, escalabilidade, confiabilidade etc;
- Análise: nessa fase é realizada a modelagem do problema, a avaliação e a documentação dos requisitos capturados na fase anterior. É válido ressaltar que nessa fase o problema é tratado de forma completamente isolada de barreiras tecnológicas (arquiteturas, bancos de dados e linguagens de programação específicas), com o intuito de facilitar uma comunicação com os usuários e manter uma isenção em relação a possíveis projetos em diferentes plataformas-alvo;
- Projeto: fase responsável por trazer o problema modelado na fase anterior para o mundo computacional, tratando, portanto, de características da arquitetura a ser utilizada, banco de dados escolhido, linguagem de programação etc;
- Implementação: nessa fase o projeto é codificado na linguagem de programação definida na fase de projeto;
- Testes: nessa fase, tipicamente, são feitos vários níveis de teste - testes de unidade, testes de integração e testes de validação. Os testes de unidade correspondem aos testes feitos com as menores unidades de software desenvolvidas. Em seguida são feitos os testes de integração, quando as várias unidades de software são colocadas para trabalharem juntas, e finalmente o sistema é testado como um todo. Neste momento, são realizados testes de validação, que visam a assegurar que todos os requisitos capturados no projeto foram implementados de forma a atenderem aos requisitos do cliente.

Neste trabalho foi definido um processo de software com as seguintes atividades: planejamento do projeto, levantamento de requisitos, análise, projeto, implementação e testes. A seguir é apresentada uma breve descrição de cada uma dessas atividades.

3.2 O Modelo de Ciclo de Vida Adotado

Inicialmente foi escolhido o modelo de ciclo de vida a ser adotado no projeto. Essa escolha considerou os seguintes fatores:

- Os trabalhos desenvolvidos em ODE estão constantemente em evolução;
- Parte do trabalho em questão é re-estruturação de ferramentas do ambiente;
- Parte do trabalho é a criação de novas ferramentas;

Tendo em vista essas circunstâncias, foi escolhido o modelo de ciclo de vida espiral (mostrado na Figura 3.1), que utiliza um processo de desenvolvimento incremental, gerando versões prototípicas, até que uma versão ganhe maturidade o bastante para ser utilizada.

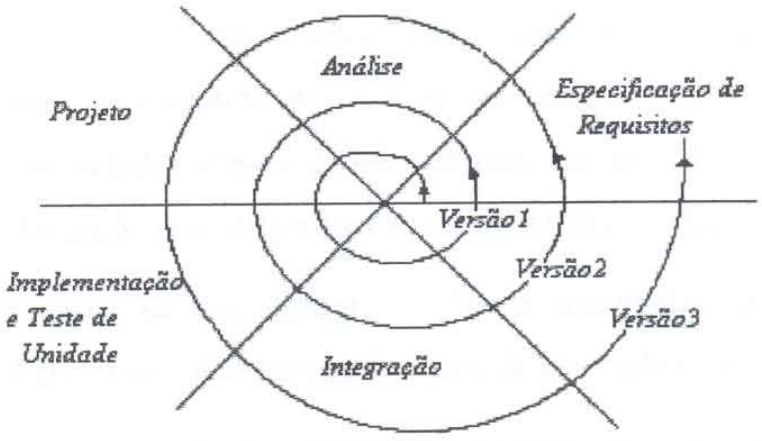

Figura 3.3 - Ciclo de vida espiral

O trabalho de re-estruturação de ferramentas gera novas versões das mesmas, enquanto o trabalho de criação de novas ferramentas concebem uma versão inicial das mesmas.

3.3 As Principais Atividades do Processo

Na fase de levantamento de requisitos o problema, juntamente com suas restrições, foi levantado e entendido. Durante essa fase foi utilizada a modelagem de caso de uso. Foram identificados atores e casos de uso do sistema e, posteriormente, foram feitas descrições de cada um deles, mostrando para cada caso de uso um conjunto de cursos normais e cursos alternativos (caso existam). No capítulo 4 encontra-se parte dos modelos e descrições de casos de uso definidos neste trabalho.

Durante a modelagem e descrição dos casos de uso do sistema, são relacionadas as possíveis classes do sistema, utilizando uma abordagem orientada a objetos. Assim, em seguida foram elaborados diagramas de classes de análise. Vale lembrar que na fase de análise não são considerados aspectos computacionais de implementação. O capítulo 5 apresenta os modelos gerados nessa fase.

Na fase de projeto, o trabalho foi re-organizado em forma de componentes levando em conta os aspectos tecnológicos. Os seguintes componentes foram levados em conta:

- Componente do Domínio do Problema: grupo de classes que modela os conceitos do domínio do problema;
- Componente de Gerência de Dados: conjunto de classes que interage diretamente sobre a camada de persistência, facilitando a persistência dos objetos do domínio;
- Componente de Interação Humana: grupo de classes responsáveis pela visão da interface com o usuário;
- Componente de Gerência de Tarefas: grupo de classes responsáveis por tratar as regras de negócio definidas pelos casos de uso;
- Componente de Controle de Interação: grupo de classes responsáveis por fazer a ligação entre as camadas de Gerência de Tarefas e de Interação Humana, isolando-as e permitindo que as funcionalidades da primeira sejam re-utilizadas quando necessário.

Para cada um dos componentes descritos anteriormente, diagramas de classes foram elaborados. Nesta monografia, parte do projeto realizado é discutido no Capítulo 6.

Com o sistema modelado sob o foco tecnológico, a implementação do mesmo pode se dar de forma mais fácil, já que todos aspectos tecnológicos foram cuidadosamente levados em conta. Na fase de implementação foi utilizada a linguagem de programação Java, adotada no Projeto ODE. A persistência de objetos foi realizada utilizando-se o framework de persistência Hibernate (BAUER et al., 2005) e o banco de dados PostgreSQL, muito utilizado pela comunidade de código aberto e até por organizações de grande porte.

Na fase de testes foram feitos casos de testes abrangendo os casos de uso definidos para o sistema. É valido lembrar que os testes não foram concebidos com o intuito de provar que o sistema nunca falha, e sim para apontar a existência de falhas.

Por fim, é importante destacar que os modelos concebidos neste trabalho nas fases de levantamento de requisitos, análise e projeto utilizaram a Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) (BOOCH et al., 2005).

Capítulo 4

Especificação de Requisitos

A Especificação de Requisitos tem como objetivo entender e descrever o problema-alvo, dando uma solução em alto-nível e se atendo apenas às solicitações do cliente. Durante essa fase é produzida, ainda, uma modelagem para as funcionalidades identificadas. Neste trabalho foi utilizada a modelagem de casos de uso para este fim.

Este capítulo apresenta os casos de uso elaborados para a ferramenta de Decomposição do Produto e Elaboração de Estruturas Analíticas de Trabalho (EATs).

As seções 4.1 e 4.2 apresentam os requisitos funcionais das ferramentas de Decomposição do Produto e Elaboração de EATs, respectivamente. Em seguida, a seção 4.3 apresenta os requisitos não-funcionais considerados neste trabalho.

É valido ressaltar que as Especificações de Requisitos das ferramentas re-estruturadas, Ferramenta de Apoio a Estimativas Usando Pontos de Função e Ferramenta de Apoio a Estimativas Usando Pontos de Caso de Uso, encontram-se, respectivamente, nos anexos A e B.

4.1 Especificação de Requisitos Funcionais – Decomposição do Produto

Nesta seção, são apresentados os requisitos funcionais, incluindo casos de uso e suas respectivas descrições, identificados para a construção da ferramenta de apoio à decomposição do produto.

4.1.1 Descrição do Mini-Mundo

Apesar da extrema importância de se ter um processo definido e tirar proveito de sua decomposição, no ambiente ODE ainda não havia sido abordado o passo inicial da fase de planejamento de um projeto: a determinação do seu escopo (PRESSMAN, 2002). Com a

determinação do escopo, a decomposição do produto passa a ser uma estratégia de apoio à gerência extremamente útil, que pode trazer inúmeros benefícios.

Na decomposição do produto podemos subdividir recursivamente o sistema em módulos. Isto é, podemos dividir um sistema em subsistemas, estes em módulos e os módulos em sub-módulos. Além disso, é possível atribuir dependências entre módulos, indicando que o desenvolvimento de um módulo depende, pelo menos parcialmente, do desenvolvimento de outros. Não é preciso indicar uma dependência entre um módulo e seus sub-módulos, já que para o desenvolvimento de um módulo é sempre necessário o desenvolvimento de seus sub-módulos.

Para cada módulo uma caracterização pode ser feita, determinando características básicas do módulo em questão, tais como: complexidade, tamanho, a equipe necessária, experiência necessária etc. Feita a caracterização de um módulo, pode-se achar outros módulos que têm algum grau de similaridade para comparação futura (estimativas, por exemplo).

A decomposição do produto em ODE é tratada no pacote *Produto*, mostrado na Figura 4.1.

Figura 4.1 – Diagrama de Pacotes.

O pacote *Produto* contempla os elementos e funcionalidades relacionadas à ferramenta de Decomposição do Produto.

4.1.2 Modelo de Casos de Uso

A Figura 4.2 mostra o diagrama de casos de uso referente ao pacote *Produto*. Na seqüência, os casos de uso identificados são descritos.

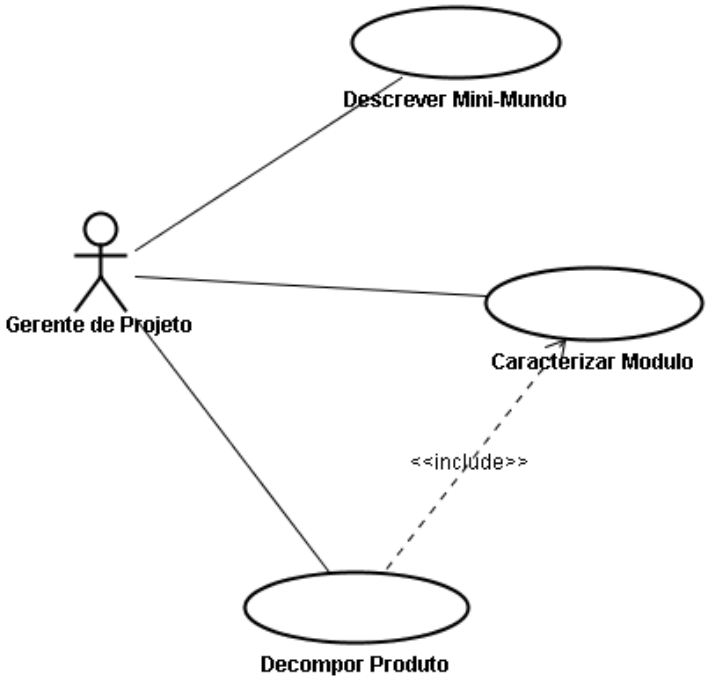

Figura 4.2 – Diagrama de Caso de Uso do Pacote *Produto*

4.1.2.1 - Caso de Uso: Descrever Mini-Mundo

Este caso de uso é responsável por descrever o escopo do projeto em questão.

Curso Normal:

O gerente de projeto informa uma descrição do escopo do projeto (descrição do mini-mundo) para o projeto corrente. A descrição é registrada.

4.1.2.2 - Caso de Uso Caracterizar Módulo

Este caso de uso é responsável por caracterizar um módulo de um projeto.

Curso Normal:

O gerente de projeto seleciona um módulo do projeto. As características já registradas são apresentadas e o gerente de projeto pode alterá-las ou informar novas características do mesmo com seus respectivos valores. A caracterização do módulo é registrada.

4.1.2.3 - Caso de Uso: Decompor Produto

Este caso de uso é responsável pela decomposição do produto descrito no escopo do software, envolvendo a criação, alteração e exclusão de módulos.

Cursos Normais:

Criar Novo Módulo

O gerente de projeto informa o nome e a descrição do módulo a ser criado e opcionalmente um módulo-pai e os módulos dos quais ele depende. Quando o módulo não é sub-módulo de outro módulo, ele é dito um sub-sistema. O novo módulo é registrado.

Alterar Dados de Módulo

O gerente de projeto informa o módulo e os novos dados. Os novos dados são validados e a alteração registrada.

Excluir Módulo

O gerente de projeto informa o módulo que deseja excluir. Os dados do módulo são apresentados e é solicitada confirmação. Se a exclusão for confirmada, o módulo é excluído juntamente com seus sub-módulos.

Curso Alternativo:

Criar Novo Módulo / Alterar Dados de Módulo

Gerente tenta definir uma dependência entre um módulo-pai e um módulo-filho: uma mensagem é apresentada, informando que não é possível definir esse tipo de dependência.

Restrições de Integridade:

Uma hierarquia de módulos deve ter todos seus componentes pertencendo ao mesmo escopo.

4.2 Especificação de Requisitos Funcionais – Elaboração de EATs

Nesta seção, são apresentados requisitos funcionais, incluindo os casos de uso, e suas respectivas descrições, identificados para a construção da ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho - EATs.

4.2.1 Descrição do Mini-Mundo

Uma Estrutura Analítica de Trabalho (EAT) é uma ferramenta de gerenciamento de projetos que contém uma série de informações que variam no nível de detalhamento desejado, em cima do escopo do projeto em questão (MARTINS, 2005).

Para se criar uma nova EAT, deve-se decompor o projeto corrente em partes menores, tentando alcançar o nível de pacotes de trabalho, sendo esses as menores unidades gerenciáveis em um projeto. Um agrupamento de pacotes de trabalho é conhecido como tarefa-resumo.

A decomposição do projeto dá-se, portanto, em três níveis:

0. Nível de Projeto → Aqui se encontra a raiz do EAT, o projeto;
1. Nível de Tarefa-Resumo → Aqui se encontram as tarefas-resumo, resultado da decomposição do projeto;
2. Nível de Pacote de Trabalho → Aqui se encontram as menores unidades gerenciáveis no projeto, os pacotes de trabalho.

Ao se incluir uma nova estrutura analítica de trabalho deve ser informado qual é o escopo da mesma, havendo, para isso, duas possibilidades: escopo de processo e escopo de produto. O escopo de uma estrutura analítica de trabalho define se as tarefas-resumo da mesma serão extraídas do processo de software (escopo de processo) ou da decomposição do produto (escopo de produto).

As tarefas-resumo têm o papel de agrupar os pacotes de trabalho encontrados numa EAT, provendo informações de sumarização relacionadas aos mesmos. Na inclusão de uma tarefa-resumo devem ser informados as atividades ou os módulos, ambos referentes ao escopo definido na criação da EAT, que a mesma vai contemplar.

O pacote de trabalho é a menor unidade gerenciável dentro de uma EAT. É nesse elemento que são guardadas as informações relativas à gerência do projeto propriamente dita. Ao se incluir um pacote de trabalho em uma tarefa-resumo, devem ser informadas as atividades ou módulos que o pacote de trabalho vai contemplar. Isso depende do escopo definido para as tarefas-resumo. Se o escopo das tarefas-resumo for o escopo de processo, o escopo dos pacotes de trabalho será de produto e, portanto, módulos deverão ser informados. Se o escopo das tarefas-resumo for escopo de produto, o escopo dos pacotes de trabalho será de processo e, portanto, atividades deverão ser informadas.

A Figura 4.3 mostra o diagrama de pacotes da ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho (EATs).

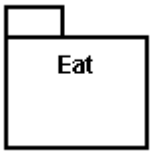

Figura 4.3 - Diagrama de Pacotes

Os casos de uso que se referem às funcionalidades da ferramenta de Elaboração de EATs são tratados no pacote `Eat`.

4.2.2 Modelo de Casos de Uso

A Figura 4.4 mostra o diagrama de casos de uso do pacote *Eat*. Vale destacar que o caso de uso *Controlar Dados Gerenciais de Pacote de Trabalho* é abstrato, representando genericamente as funcionalidades para incluir, alterar ou excluir dados gerenciais em pacotes de trabalho. Neste trabalho, apenas dados de estimativas de esforço foram tratados e, portanto, há apenas uma especialização desse caso de uso. A seguir, os casos de uso mostrados na figura são descritos.

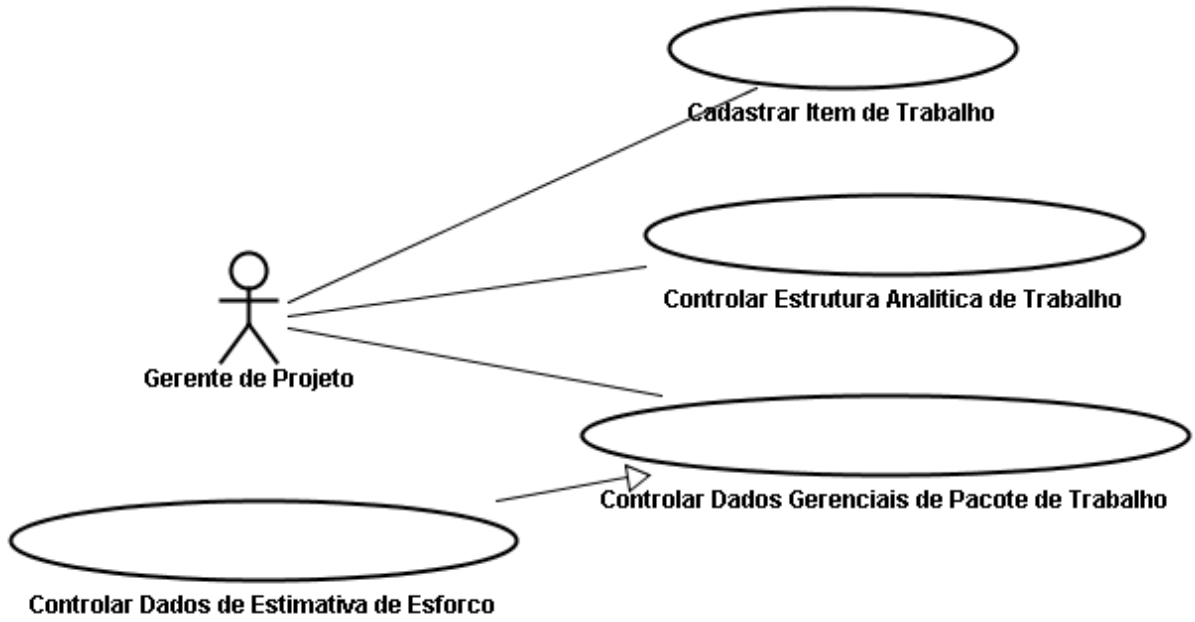

Figura 4.4 - Diagrama de Casos de Uso do pacote Eat

4.2.2.1 - Caso de Uso Controlar Estrutura Analítica de Trabalho

Este caso de uso é responsável pela inclusão, alteração, visualização e exclusão de uma EAT.

Cursos Normais:

Incluir Nova EAT:

O gerente informa o nome, a descrição e o escopo das tarefas-resumo da EAT (processo ou produto) e uma nova EAT é criada para o projeto corrente, tendo um item de trabalho raiz com o nome e a descrição informados. A data de criação da EAT é também registrada.

Excluir EAT:

O gerente informa a EAT que deseja excluir. Uma mensagem de confirmação é exibida e, se confirmada, a EAT juntamente com seus itens de trabalho relacionados são excluídos.

Visualizar Estrutura Analítica de Trabalho:

O gerente de projeto informa a EAT que deseja visualizar e a mesma é apresentada, sendo que, para cada tarefa resumo, é exibida uma sumarização dos dados gerenciais registrados em cada pacote de trabalho.

4.2.2.2 - Caso de Uso: Cadastrar Item de Trabalho

Este caso de uso é responsável pelo controle dos itens de trabalho em uma EAT.

Cursos Normais:

Incluir Nova Tarefa-Resumo:

O gerente de projeto informa o nome e a descrição da tarefa-resumo e seleciona os módulos ou macro-atividades que esta vai contemplar, de acordo com escopo definido na criação da EAT (caso de uso *Controlar Estrutura Analítica de Trabalho*, evento *Incluir Nova EAT*), respectivamente produto ou processo. Os dados são validados e registrados.

Incluir Novo Pacote de Trabalho:

O gerente de projeto informa o nome e a descrição do Pacote de Trabalho e seleciona as atividades ou módulos que o mesmo vai contemplar, de acordo com o escopo das tarefas-resumo escolhido na criação da EAT. Se o escopo das tarefas-resumo for o escopo de processo, o escopo dos pacotes de trabalho será de produto e, portanto, módulos deverão ser informados. Se o escopo das tarefas-resumo for escopo de produto, o escopo dos pacotes de trabalho será de processo e, portanto, atividades deverão ser informadas. Os dados são validados e registrados.

Alterar Item de Trabalho:

O gerente de projeto informa os novos dados do item de trabalho. Os dados são validados e registrados.

Excluir Item de Trabalho:

O gerente de projeto informa o item de trabalho que deseja excluir. Uma mensagem de confirmação é exibida. O item de trabalho é excluído do sistema, juntamente com seus sub-itens (no caso de tarefas-resumo) e informações gerenciais relacionadas. Não é permitida a exclusão do item de trabalho raiz da EAT.

4.2.2.3 - Caso de Uso Controlar Dados Gerenciais de Pacote de Trabalho

Este caso de uso é responsável pela inclusão, alteração e exclusão de dados gerenciais em um pacote de trabalho.

Cursos Normais:

Incluir Novo Dado Gerencial em Pacote de Trabalho:

O gerente informa o pacote de trabalho, o dado gerencial que deseja inserir e o valor relacionado ao mesmo. Os dados são validados e registrados no sistema.

Alterar Dado Gerencial em Pacote de Trabalho:

O gerente informa o pacote de trabalho, o dado gerencial que deseja alterar e o novo valor referente ao mesmo. Os dados são validados e registrados no sistema.

Excluir Dado Gerencial em Pacote de Trabalho:

O gerente informa o pacote de trabalho e o dado gerencial que deseja excluir. Uma mensagem de confirmação é exibida. O dado gerencial é excluído do pacote de trabalho.

4.3.2.4 - Caso de Uso Controlar Dados de Estimativa de Esforço

Este caso de uso é responsável pela inclusão, alteração e exclusão de estimativas de esforço em um pacote de trabalho, sendo, portanto, uma especialização do caso de uso Controlar Dados Gerenciais de Pacote de Trabalho.

Cursos Normais:

Conforme descritos no caso de uso “Controlar Dados Gerenciais de Pacote de Trabalho”, sendo os dados gerenciais em questão valores de estimativas de esforço, é necessário informar apenas o valor da estimativa.

4.3 Especificação de Requisitos Não-Funcionais

Nesta seção, são apresentados os requisitos não-funcionais identificados como mais importantes para este trabalho.

Os requisitos ditos não-funcionais são tratados na fase de implementação e devem satisfazer alguns critérios de qualidade que um sistema precisa atender, dentre eles (MEYER, 1997):

- Facilidade de manutenção (manutenibilidade).
- Uso otimizado dos recursos computacionais (desempenho);
- Funcionamento sob condições anormais (robustez ou escalabilidade);
- Facilidade de operar o sistema (facilidade de uso);
- Facilidade de integração com outros produtos de software (compatibilidade);
- Preservação da disponibilidade e da integridade das informações armazenadas (confiabilidade);
- Proteção contra acessos indevidos (segurança).

A maior parte das ferramentas CASE encontradas no ambiente ODE atendem aos critérios supra-citados. As ferramentas propostas nesse trabalho não são exceção, porém é importante ressaltar alguns critérios que se sobressaem perante os outros, a saber: manutenibilidade, facilidade de uso e compatibilidade.

4.3.1 Manutenibilidade

Manutenibilidade é um critério de extrema importância para qualquer software, pois mudanças são inevitáveis.

Nas ferramentas descritas neste trabalho foi utilizada uma nova arquitetura de software (detalhada na seção 6.1), com o objetivo de permitir fazer manutenções nas interfaces das ferramentas, sem modificação alguma nas regras de negócio ou domínio do problema.

4.3.2 Facilidade de Uso

Um software que apresenta uma interface complexa e que ainda não facilita o uso no lado do cliente está fadado ao desuso, já que a quantidade de sistemas com um mesmo propósito cresce cada vez mais, permitindo que um usuário escolha um outro sistema com a interface mais amigável e que atenda às funcionalidades requisitadas.

Neste trabalho foi utilizado um novo padrão de interface para as ferramentas que seguem um fluxo de trabalho bem definido, provendo um fluxo natural das funcionalidades encontradas em cada uma delas. Além disso, as funcionalidades implementadas são tratadas de forma mais clara, utilizando-se de mensagens explicativas, painéis e outros elementos de interfaces gráficas para facilitar o uso de cada ferramenta.

4.3.2 Compatibilidade

Todo ADS deve ter como principal propósito a integração entre as ferramentas que o compõem, provendo facilidade de troca de informações.

Nas ferramentas descritas neste trabalho, foi feito um esforço no sentido de prover uma alta integração entre elas, de forma que os dados compartilhados entre as ferramentas possam ser visualizados e até mesmo alterados tanto por uma quanto por outras.

Capítulo 5

Análise

Na fase de análise o problema descrito na Especificação de Requisitos é estudado com mais profundidade, tentando elaborar um modelo que represente o domínio do sistema. Em especial na Análise Orientada a Objetos são identificados os objetos, seus relacionamentos e comportamentos necessários para a construção do mesmo.

Este capítulo apresenta os diagramas de classes elaborados para as ferramentas de Decomposição do Produto e Elaboração de EATs nas seções 5.1 e 5.2, respectivamente.

É válido ressaltar que a Especificação de Análise dos projetos de re-estruturação das ferramentas de Estimativas de Pontos de Função e de Pontos de Caso de Uso são encontradas nos anexos A e B, respectivamente.

5.1 Modelagem de Classes – Decomposição do Produto

A modelagem de classes envolve a identificação de classes, atributos, associações e operações, bem como o agrupamento de classes em subsistemas ou pacotes. Na Figura 5.1 é apresentado o diagrama de pacotes referente à porção do sistema que trata da decomposição de produtos de software.

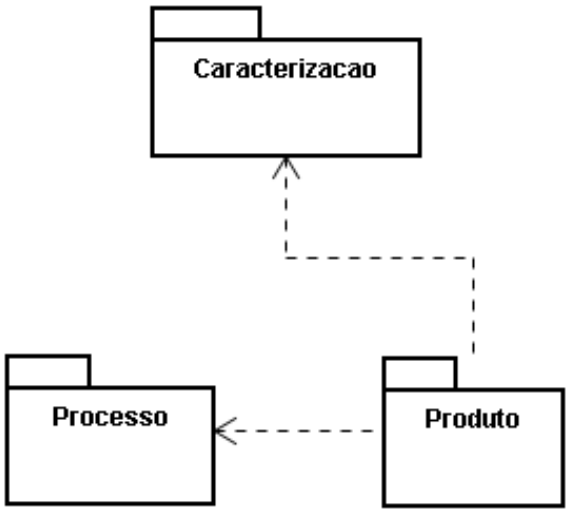

Figura 5.1 – Diagrama de Pacotes da ferramenta de apoio à Decomposição do Produto.

Conforme discutido no capítulo anterior, no pacote *Processo* encontra-se parte do núcleo do ambiente ODE, como a classe `Projeto`. O maquinário de caracterização, contido no pacote *Caracterizacao*, trata da caracterização de elementos de ODE (projetos, atividades, módulos etc), permitindo comparação futura entre os mesmos. O pacote *Produto* contempla os elementos e funcionalidades relacionados à ferramenta de Decomposição do Produto, que necessita de classes tanto do pacote *Processo* quanto do pacote *Caracterizacao* para cumprir suas responsabilidades.

A Figura 5.2 apresenta o diagrama de classes do pacote *Produto*, foco deste trabalho. Um projeto pode ter seu escopo definido (classe `Escopo`). Cada escopo é composto de módulos, representados pela classe `Modulo`, que podem ser caracterizados (classe `Caracterizacao`), permitindo que os mesmos possam ser comparados. Um módulo pode, ainda, ser associado a uma execução de módulo (classe `ExecucaoModulo`), que é a via de conexão de um módulo com outros elementos do ambiente ODE, como estimativas, alocações etc.

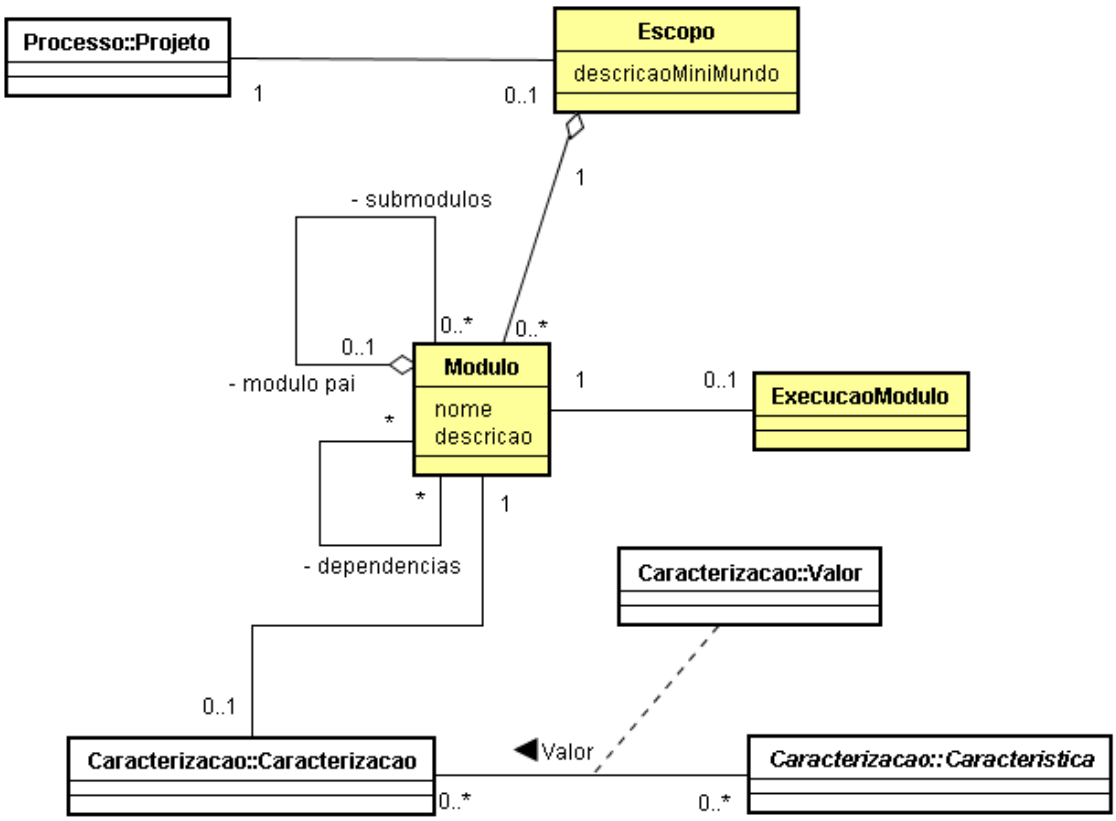

Figura 5.2 – Diagrama de Classes do pacote *Produto*

A comparação entre módulos caracterizados tem o objetivo de auxiliar gerentes de projeto a encontrarem módulos similares para, por exemplo, estimar o esforço necessário para o desenvolvimento de um módulo a partir do esforço despendido para desenvolver um módulo similar.

5.2 Modelagem de Classes – Elaboração de EATs

O diagrama de pacotes da Figura 5.3 mostra as dependências existentes entre os pacotes utilizados na modelagem de classes da ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho (EATs). A figura 5.4, por sua vez, mostra o diagrama de classes do pacote *Eat*, foco deste trabalho.

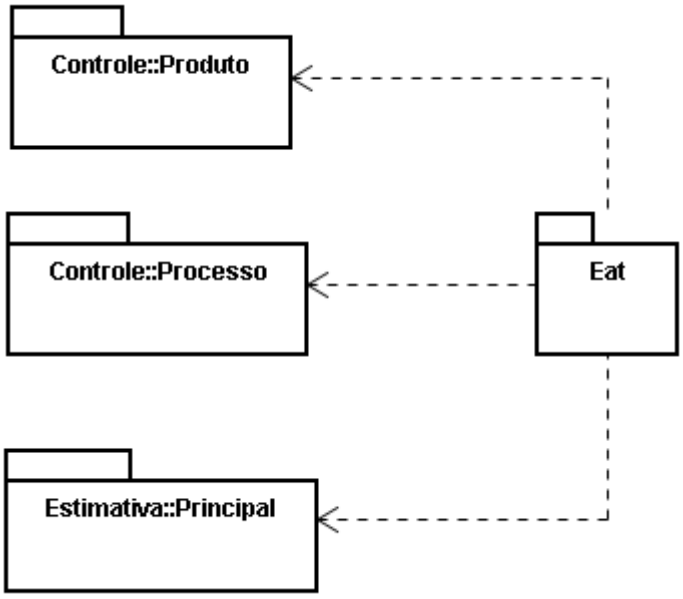

Figura 5.3 - Diagrama de Pacotes da Ferramenta de Apoio à Elaboração de EATs.

A classe `ItemTrabalho` representa os elementos encontrados em uma EAT. Quando um item de trabalho não é parte de nenhum outro item de trabalho, tem-se o item de trabalho raiz da EAT que representa o projeto como um todo. A relação de um item de trabalho com uma ou mais atividade(s) ou com um ou mais módulo(s) depende do escopo da EAT (processo ou produto) definido na classe `EstruturaAnaliticaTrabalho`. Se o escopo de processo for escolhido, as tarefas-resumo dessa EAT contemplarão as atividades do processo, enquanto os pacotes de trabalho se relacionarão com os módulos da decomposição do produto. Em qualquer caso, não é permitido um relacionamento entre um item de trabalho e objetos das classes `Modulo` e `Atividade` simultaneamente.

Uma tarefa-resumo é um item de trabalho que é decomposto em outros itens de trabalho (pacotes de trabalho) e está subordinado diretamente ao item de trabalho raiz (aquele que representa o projeto na EAT). Um pacote de trabalho é a menor parte gerenciável de uma EAT e pode contemplar atividades ou módulos (dependendo do escopo da EAT, conforme descrito anteriormente). Além disso, o pacote de trabalho pode comportar dados gerenciais quaisquer. Inicialmente está sendo contemplado apenas o registro de estimativas relacionadas a pacotes de trabalho, mais especificamente estimativas

de esforço. A classe `ExecucaoItemTrabalho` faz a ligação entre o pacote de trabalho e a estimativa-alvo.

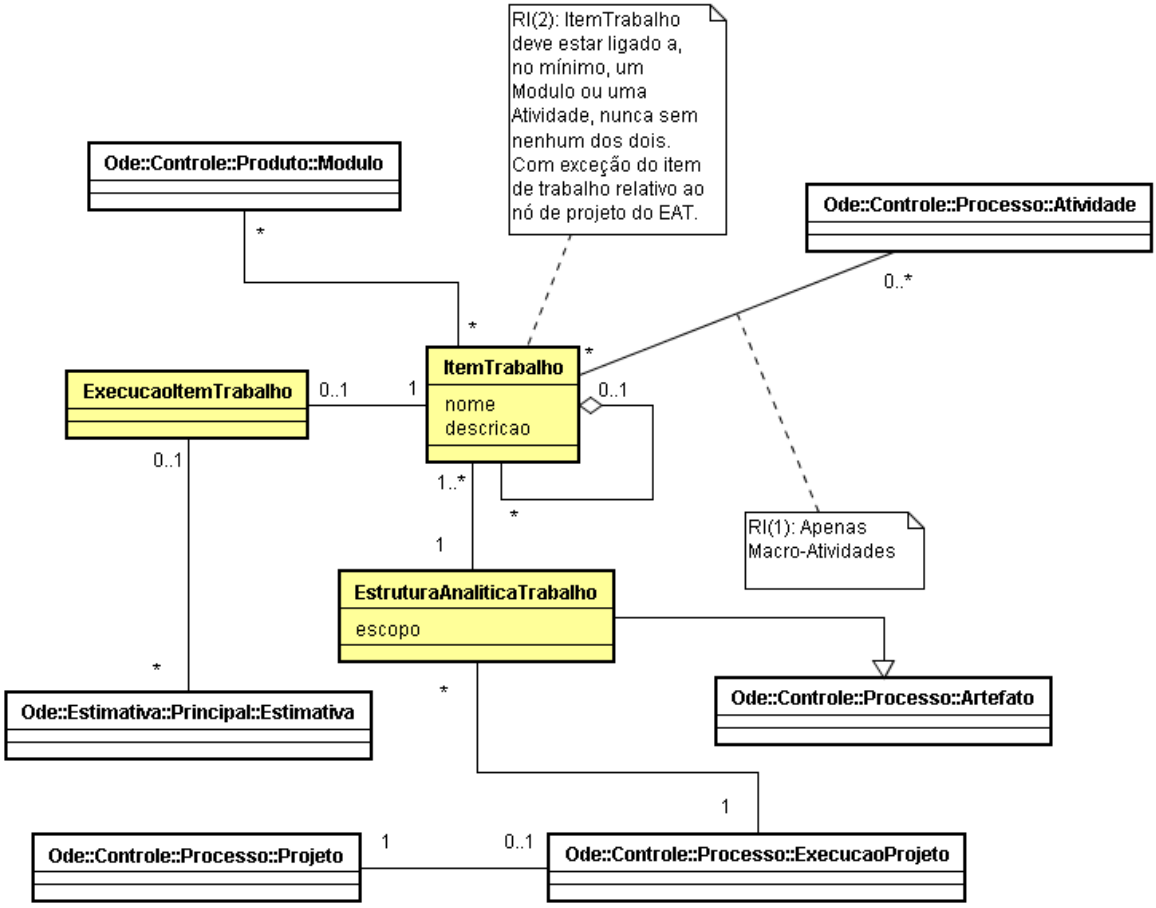

Figura 5.4 - Diagrama de Classes do pacote *Eat*.

Capítulo 6

Projeto, Implementação e Testes

Durante a fase de análise, o sistema é modelado sem levar em conta os aspectos tecnológicos a serem utilizados para materializá-lo. A fase de projeto tem o objetivo de definir todos os aspectos tecnológicos do sistema, tais como linguagem de programação utilizada na implementação, arquitetura do software, características de interface com o usuário e formas de acesso e persistência de dados.

Seguindo os padrões utilizados no Projeto ODE, a implementação é feita na linguagem de programação Java, utilizando um banco de dados relacional com o *framework* de persistência Hibernate (BAUER et al., 2005). A nova arquitetura de software adotada no projeto é composta de cinco componentes, a saber: Componente de Domínio do Problema (Cdp), Componente de Gerência de Dados (Cgd), Componente de Gerência de Tarefas (Cgt), Componente de Interação Humana (Cih) e Componente de Controle de Interação (Cci).

Nas partes deste trabalho que envolveram reestruturação, a saber: as ferramentas de estimativa de pontos de função e pontos de caso de uso, o projeto foi efetuado a partir dos trabalhos existentes, adequando-os à nova arquitetura, além de se adicionar as novas funcionalidades propostas no documento de especificação de requisitos e adequar essas ferramentas a um novo padrão de interface com o usuário proposto recentemente. Para maiores detalhes sobre essas ferramentas, vide anexos A e B, respectivamente.

Neste capítulo, apresentamos a parte do projeto, implementação e testes das duas novas ferramentas desenvolvidas neste trabalho: Ferramenta de Decomposição do Produto e Ferramenta de apoio à Elaboração de Estruturas Analíticas de Trabalho (EATs).

Na seção 6.1 discutimos sobre a nova arquitetura de software proposta. Dando continuidade, a seção 6.2 disserta sobre a nova camada de persistência utilizada e sobre os padrões que foram definidos para o bom funcionamento da mesma. A seção 6.3 apresenta os utilitários de persistência definidos para o ambiente ODE. Nas seções 6.4 e 6.5, são

apresentados, respectivamente, os projetos das ferramentas de Decomposição do Produto e Elaboração de EATs. A seção 6.6 comenta sucintamente o funcionamento dessas ferramentas. Por fim, a seção 6.7 discute brevemente como foram conduzidos os testes.

6.1 Projeto de Arquitetura do Sistema

O projeto da arquitetura é a primeira tarefa da fase de Projeto. Nela são definidas a organização e as dependências dos componentes do sistema-final, levando-se em conta fatores técnicos (linguagem, plataforma etc) e requisitos não funcionais.

Inicialmente o domínio do problema e todas suas classes definidas na fase de análise são avaliados quanto à necessidade da reprodução das mesmas na fase de projeto. Aspectos tecnológicos ligados à linguagem, por exemplo, podem ser definitivos na manutenção de algumas classes de análise e na criação de novas classes na fase de projeto. Com o domínio do problema de projeto definido, é necessário estabelecer como serão tratados os casos de uso definidos na documentação de requisitos.

O ambiente ODE conta, na maior parte das suas ferramentas, com uma arquitetura organizada em quatro grandes componentes, mostrada na Figura 6.1. Essa arquitetura garante que os objetos do componente de interação humana (Cih), responsável por tratar as interfaces que capturam dados do usuário ou exibem informações para o mesmo, possam enviar e receber dados das classes que realizam as regras de negócio (Componente de Gerência de Tarefas - Cgt) extraídas dos casos de uso da especificação de requisitos. O componente de domínio do problema (Cdp) contempla as classes extraídas da fase de análise e que, agora, ganham acesso às funcionalidades do mecanismo de persistência definidos nas classes do componente de gerência de dados (Cgd).

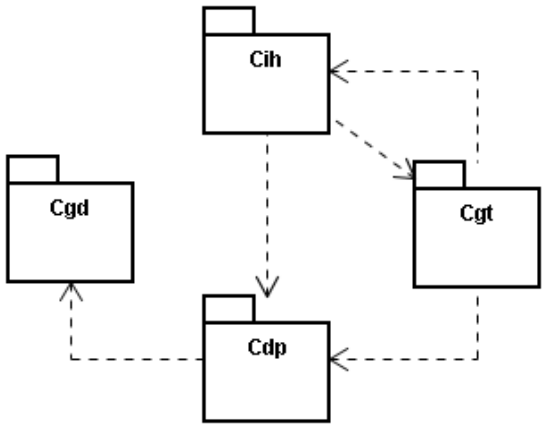

Figura 6.4 - Arquitetura de 4 Camadas

Apesar do uso efetivo dessa arquitetura nos projetos realizados no contexto de ODE, foram notadas algumas desvantagens da mesma, a saber:

- Com o Cih atrelado ao Cgt, a utilização de uma mesma interface ou funcionalidade em outro ponto do ambiente é tarefa extremamente complexa;
- Constantes manutenções na interface do ambiente ODE acarretavam mudanças também no Cgt, dado o grande acoplamento existente entre os dois;
- Tratar as funcionalidades do mecanismo de persistência via domínio do problema torna as classes do Cdp mais complexas, pois eram adicionadas operações que eram diretamente repassadas ao Cgd.

Visando minimizar esses problemas, foi proposta uma nova arquitetura de cinco camadas, introduzindo um novo componente, o Componente de Controle de Interação (Cci), para desacoplar o Cih do Cgt, permitindo que as funcionalidades da gerência de tarefas possam ser reutilizadas em outros pontos do ambiente. A Figura 6.2 mostra a nova arquitetura, utilizada neste trabalho.

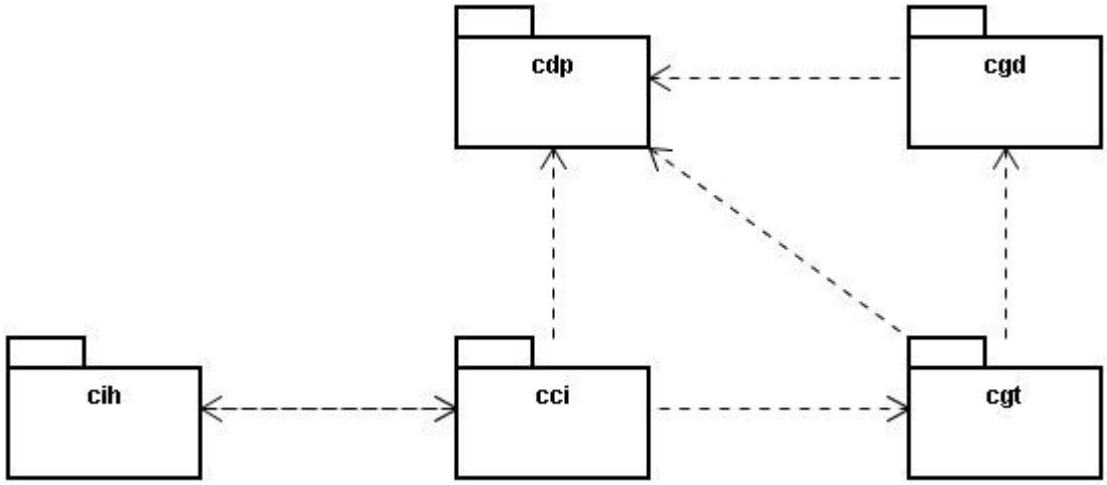

Figura 6.5 - Arquitetura de 5 camadas

A arquitetura proposta permite, também, um ambiente multi-interface, já que não é necessário um componente de gerência de tarefas para cada tipo de interface e sim uma nova camada de interface atrelada a um controlador de interface especializado para essa interface (Cci). Além disso, as classes da Cdp ficam mais simples e têm apenas operações de obtenção e atribuição de dados relacionados aos seus atributos, ficando o Cgt responsável pela comunicação com a gerência de dados.

6.2 Camada de Persistência e Padrões Utilizados

Camadas de persistência têm o intuito de agrupar um conjunto de funcionalidades e padrões que deixam o acesso ao mecanismo de persistência utilizado o mais transparente possível, sem deixar de prover flexibilidade ao sistema.

No desenvolvimento da nova camada de persistência de ODE, foram utilizados os padrões Objeto de Acesso a Dados (*Data Access Object* - DAO) (SUN, 2001) e Fábrica Abstrata (GAMMA et al., 1995), além do *framework* de persistência Hibernate (BAUER et al., 2005), descritos a seguir.

6.2.1 Hibernate

Hibernate (BAUER et al., 2005) é um *framework* gratuito de mapeamento objeto / relacional que tem o intuito de facilitar a construção de aplicações Java que dependem total

ou parcialmente de acesso a bancos de dados relacionais. Hibernate contempla, ainda, a reprodução completa das classes de projeto (com todos seus relacionamentos) em tabelas, livrando o desenvolvedor da criação de modelos relacionais. Além disso, o *framework* provê o uso de transações, permitindo que acessos concorrentes a um elemento do banco e exceções sejam tratados em tempo de execução.

Para adequar o uso do Hibernate aos padrões de implementação utilizados em ODE foram utilizados os padrões de projeto DAO e Fábrica Abstrata, com o objetivo de prover flexibilidade ao ambiente.

6.2.2 DAO – *Data Access Object*

O padrão DAO (SUN, 2001) provê uma interface comum entre aplicações e os mecanismos de persistência (banco de dados, arquivo, memória etc) e, portanto, o padrão torna-se parte integrante da camada de persistência.

O fundamento principal do padrão DAO é agrupar operações comuns relacionadas a uma classe em uma interface, que será utilizada pelas aplicações do sistema. Dessa forma, para cada mecanismo de persistência utilizado no sistema, tem-se uma implementação dessa interface. A figura 6.3 mostra o funcionamento do padrão de projeto.

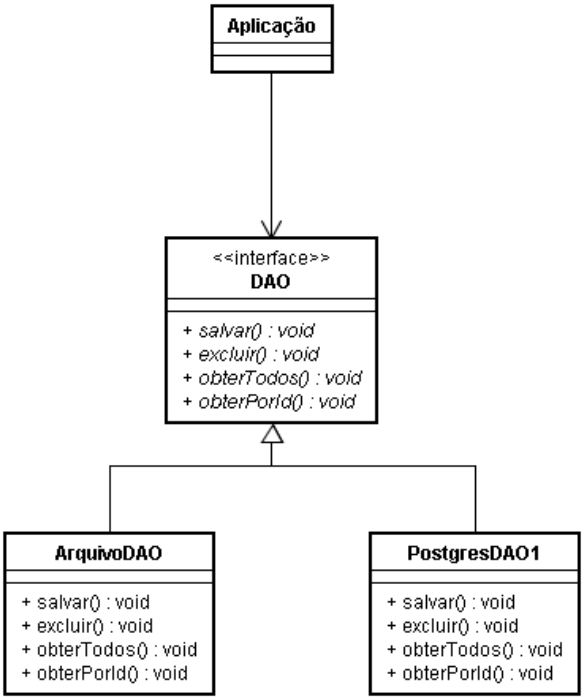

Figura 6.6 - Padrão de Projeto DAO

O padrão DAO já vinha sendo utilizado no ambiente (com o nome de classes sombra), mas seu uso estava intimamente ligado à camada de persistência antiga (RUY, 2003) e também à arquitetura de quatro camadas. Dessa forma, foi necessário adequar o padrão à nova arquitetura. Essa adequação é discutida na seção 6.3.

6.2.3 Fábrica Abstrata (*Abstract Factory*)

Uma Fábrica Abstrata (GAMMA et al., 1995) é usada neste trabalho para trabalhar diretamente com o padrão DAO e garantir que a aplicação tenha o objeto de acesso ao mecanismo de persistência apropriado para realizar suas operações, garantindo a instanciação dos mesmos de acordo com a necessidade, como ilustra a Figura 6.4. No exemplo dessa figura, cada implementação da fábrica (`ArquivoDAOFactory` e `PostgresDAOFactory`) faz a instanciação do DAO necessário à aplicação

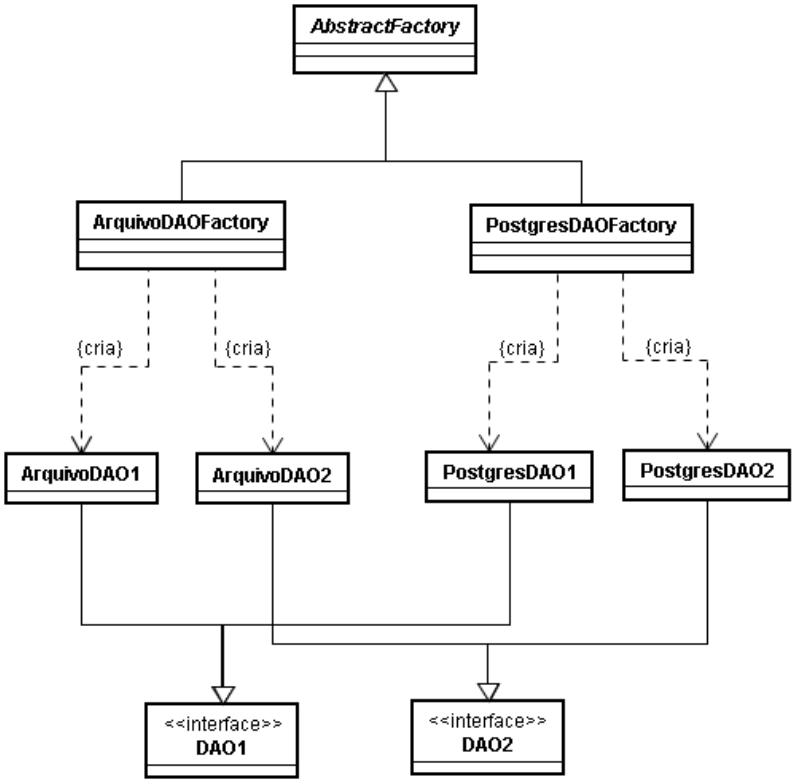

Figura 6.7 - O Padrão de Projeto *Abstract Factory*

.

6.3 Utilitários de Persistência

Nesta seção é mostrada a abordagem utilizada em ODE quanto aos padrões de projeto discutidos e quanto ao *framework* Hibernate. A Figura 6.5 mostra o pacote *Utilitario::Persistência::hibernate* definido no ambiente.

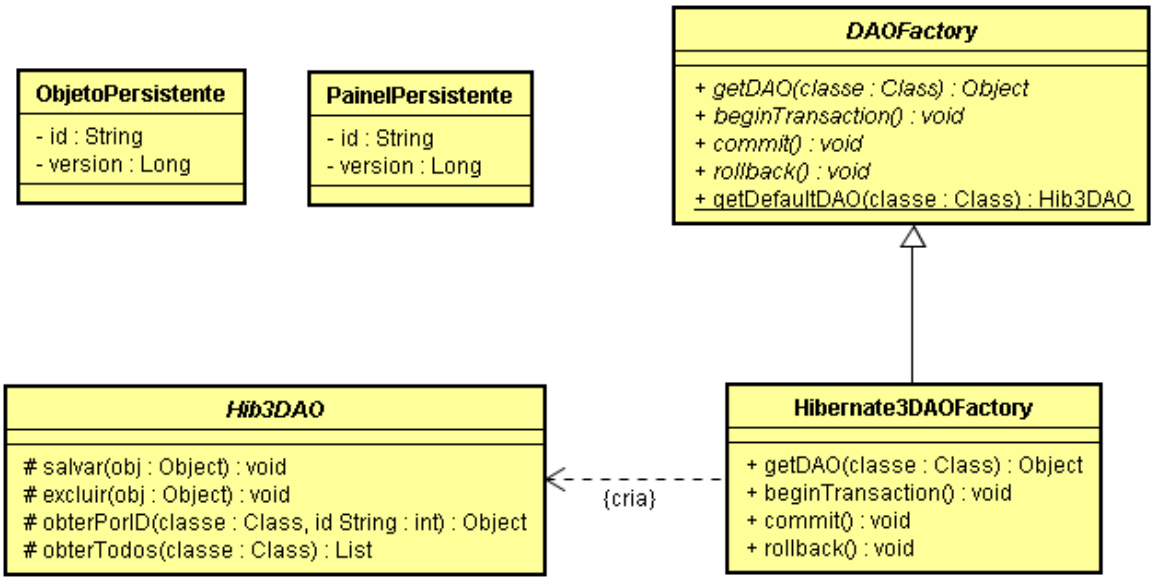

Figura 6.8 - Pacote *Utilitario::Persistencia::hibernate*

Classes de domínio e itens gráficos que devem ser persistidos devem herdar diretamente de `ObjetoPersistente` e `PainelPersistente`, respectivamente. Cada DAO relacionado herda de `Hib3DAO`, que contempla as funcionalidades básicas de acesso ao mecanismo de persistência e implementa a interface DAO associada, dando flexibilidade no momento da criação de novas operações especializadas para um certo elemento de domínio.

A operação estática de `DAOFactory`, *getDefaultDAO(classe : Class),* é utilizada para obter, a partir de uma fábrica `Hibernate3DAOFactory`, objetos de acesso ao banco via Hibernate.

6.4 Ferramenta de Decomposição do Produto

O diagrama de pacotes da Figura 6.6 mostra as dependências existentes entre os pacotes físicos contemplados nessa ferramenta.

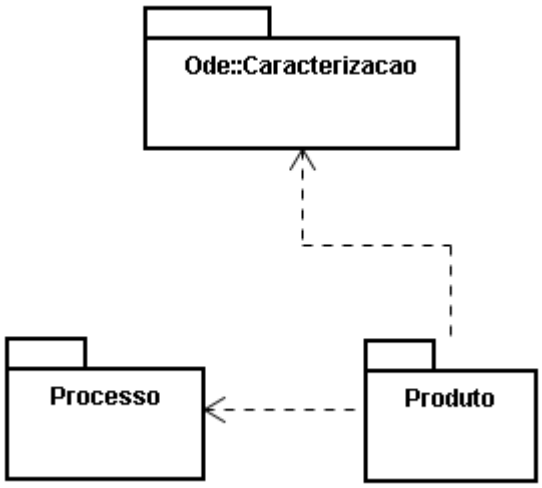

Figura 6.6 - Diagrama de pacotes

Como já foi discutido, o pacote *Produto* possui as classes que materializam a ferramenta de Decomposição do Produto no ambiente e, a seguir, são apresentados os diagramas de classes da fase de projeto desse pacote, conforme a nova arquitetura de cinco camadas de ODE.

6.4.1 Componente do Domínio do Problema (Cdp)

As classes do Cdp do pacote *Produto*, mostrado na Figura 6.7, foram originadas a partir do modelo de análise, adequando as mesmas à tecnologia Java, usada na implementação.

Não houve mudanças significativas entre os modelos de classes das fases de análise e projeto, exceto pela adição de tipos de dados de atributos e navegabilidades, e pela transformação da classe associativa Valor em uma classe comum. Essa última classe, porém, pertence ao pacote *Caracterizacao::Cdp* que não é o foco deste trabalho.

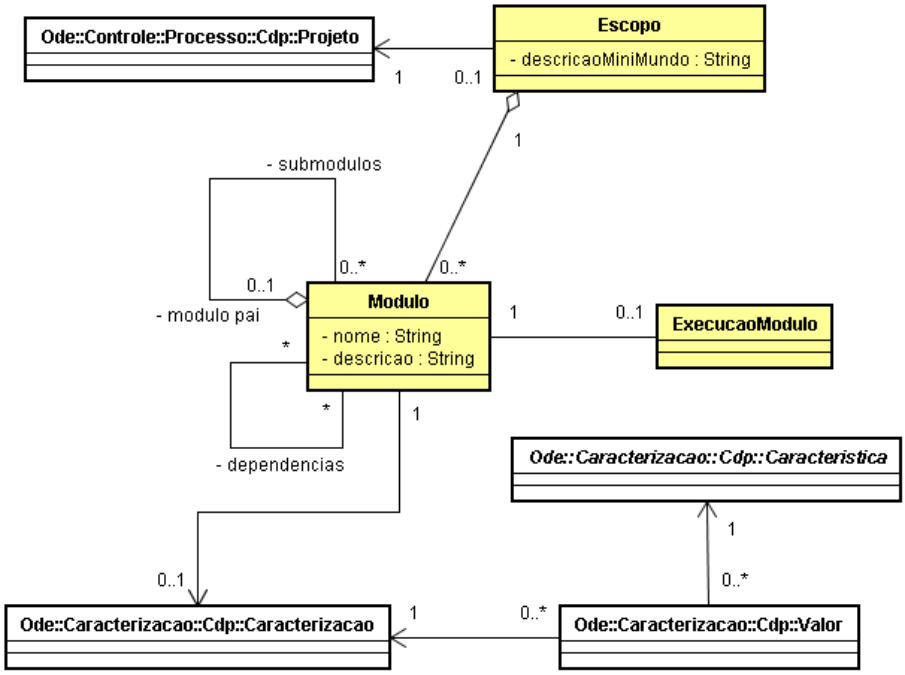

Figura 6.7 – Componente de Domínio do Problema do pacote *Produto*

6.4.2 Componente de Gerência de Dados (Cgd)

As classes contempladas no Cgd têm a função de abstrair o acesso ao banco de dados usado na implementação do trabalho. A Figura 6.8 mostra o diagrama de classes desse pacote.

Todas as classes mostradas na Figura 6.8 herdam da classe abstrata `Hib3DAO` que já implementa métodos básicos de manipulação de objetos, tais como *salvar*, *excluir*, *obterPorId* e *obterTodos*, conforme discutido na seção 6.3.

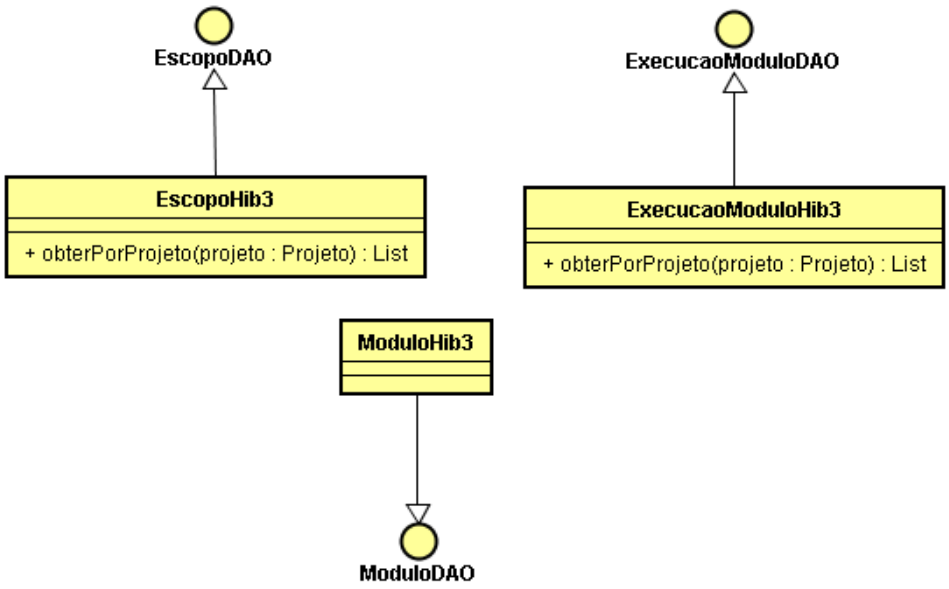

Figura 6.8 - Componente de Gerência de Dados do pacote *Produto*

6.4.3 Componente de Gerência de Tarefas (Cgt)

O componente de Gerência de Tarefas desse pacote possui uma única classe, que é responsável por realizar todos os casos de uso definidos no documento de especificação de requisitos, como mostra a Figura 6.9.

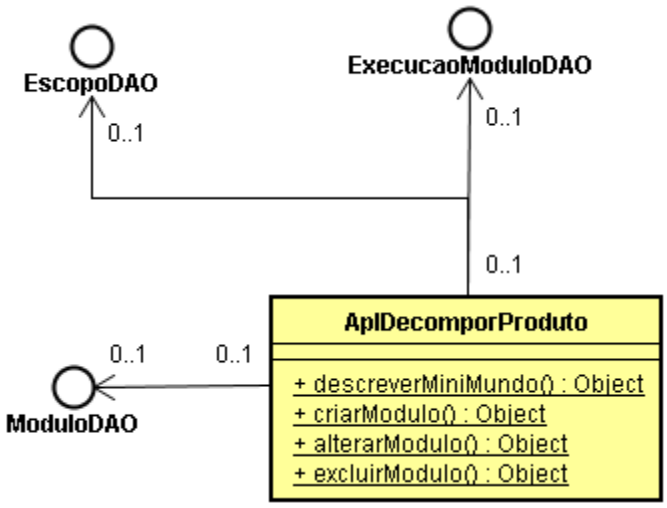

Figura 6.9 - Componente de Gerência de Tarefas do Pacote Produto

O caso de uso *Caracterizar Modulo* não foi contemplado nesta versão da ferramenta e, portanto, não há operação relativa a ele no modelo da Figura 6.9.

Os métodos implementados são estáticos e podem ser acessados sem a necessidade de instanciar objetos da classe de aplicação, provendo facilidade aos desenvolvedores que utilizam essas operações.

6.4.4 Componente de Interação Humana (Cih)

O componente de Interação Humana do pacote *Produto* materializa as interfaces com o usuário para a realização dos casos de uso definidos no documento de especificação de requisitos. A Figura 6.10 mostra o diagrama de classes desse componente.

A janela principal da ferramenta (`JanDecomporProduto`) permite que o usuário faça uma descrição do mini-mundo para o escopo do projeto ou que decomponha o escopo em módulos. No caso da descrição do mini-mundo a classe `PainelEdicaoMiniMundo` permite que o usuário descreva o escopo do produto. A classe `PainelArvoreModulo` mostra uma árvore representando a hierarquia de módulos cadastrados, sendo a raiz dessa árvore o próprio escopo do projeto. Quando um usuário deseja decompor o produto em módulos, ele usa a classe `PainelDecomporProduto` que, com um nó selecionado na árvore, permite que sejam definidos nome, descrição e módulos dependentes do módulo que está sendo criado ou alterado.

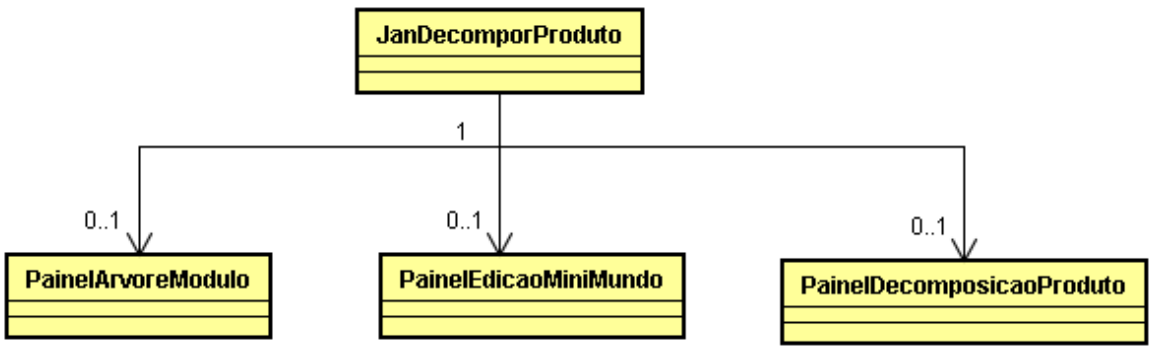

Figura 6.10 - Componente de Interface Humana do pacote *Produto*

6.4.5 Componente de Controle de Interação (Cci)

As classes do pacote Cci são responsáveis por fazer o intercâmbio de dados entre as classes de aplicação e as classes de interface com o usuário da ferramenta, como mostra a Figura 6.11.

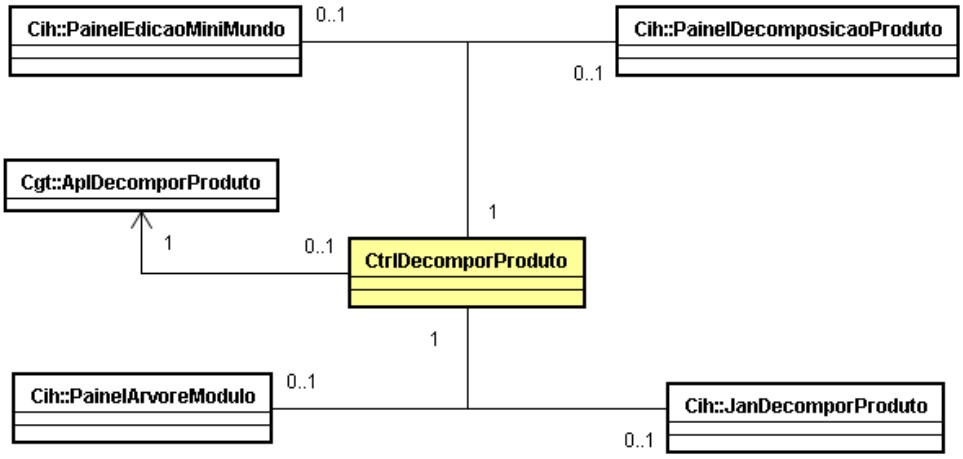

Figura 6.11 - Componente de Controle de Interação do Pacote *Produto*

Toda vez que um painel necessita acessar alguma funcionalidade, provida por uma classe de aplicação (`AplDecomporProduto`), ele o faz via `CtrlDecomporProduto`, que conhece as operações necessárias para atender à requisição da interface. Quando a aplicação requisitada retorna um resultado para o controlador, esse interpreta o mesmo e atualiza as interfaces de acordo com a necessidade.

6.5 Ferramenta de Apoio à Elaboração de EATs

O diagrama de pacotes da Figura 6.12 mostra as dependências existentes entre os pacotes físicos contemplados na ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho (EATs).

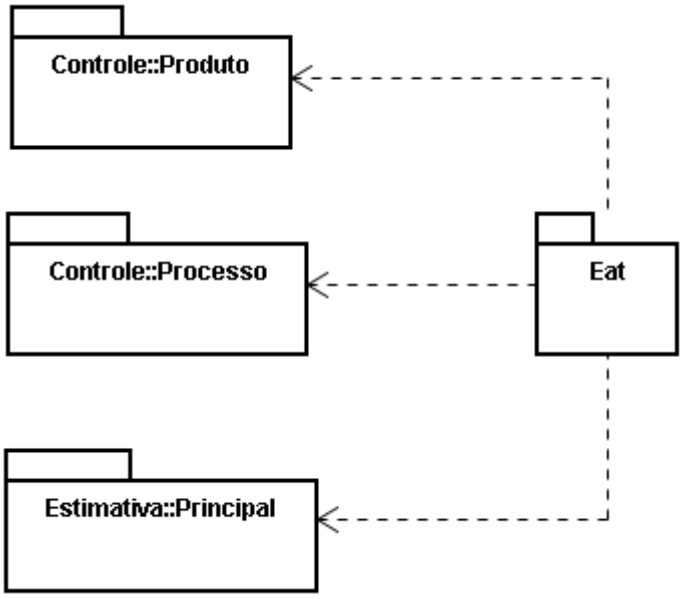

Figura 6.12 - Diagrama de pacotes

Como já foi discutido, o pacote *Eat* possui as classes que materializam a ferramenta de apoio à elaboração de Estruturas Analíticas de Trabalho no ambiente. A seguir são apresentados os diagramas de classes da fase de projeto, segundo a nova arquitetura de cinco camadas de ODE.

6.5.1 – Componente do Domínio do Problema (Cdp)

As classes do Cdp do pacote *Eat,* mostrado na Figura 6.13, foram originadas a partir do modelo de análise, adequando as mesmas à tecnologia Java, usada na implementação.

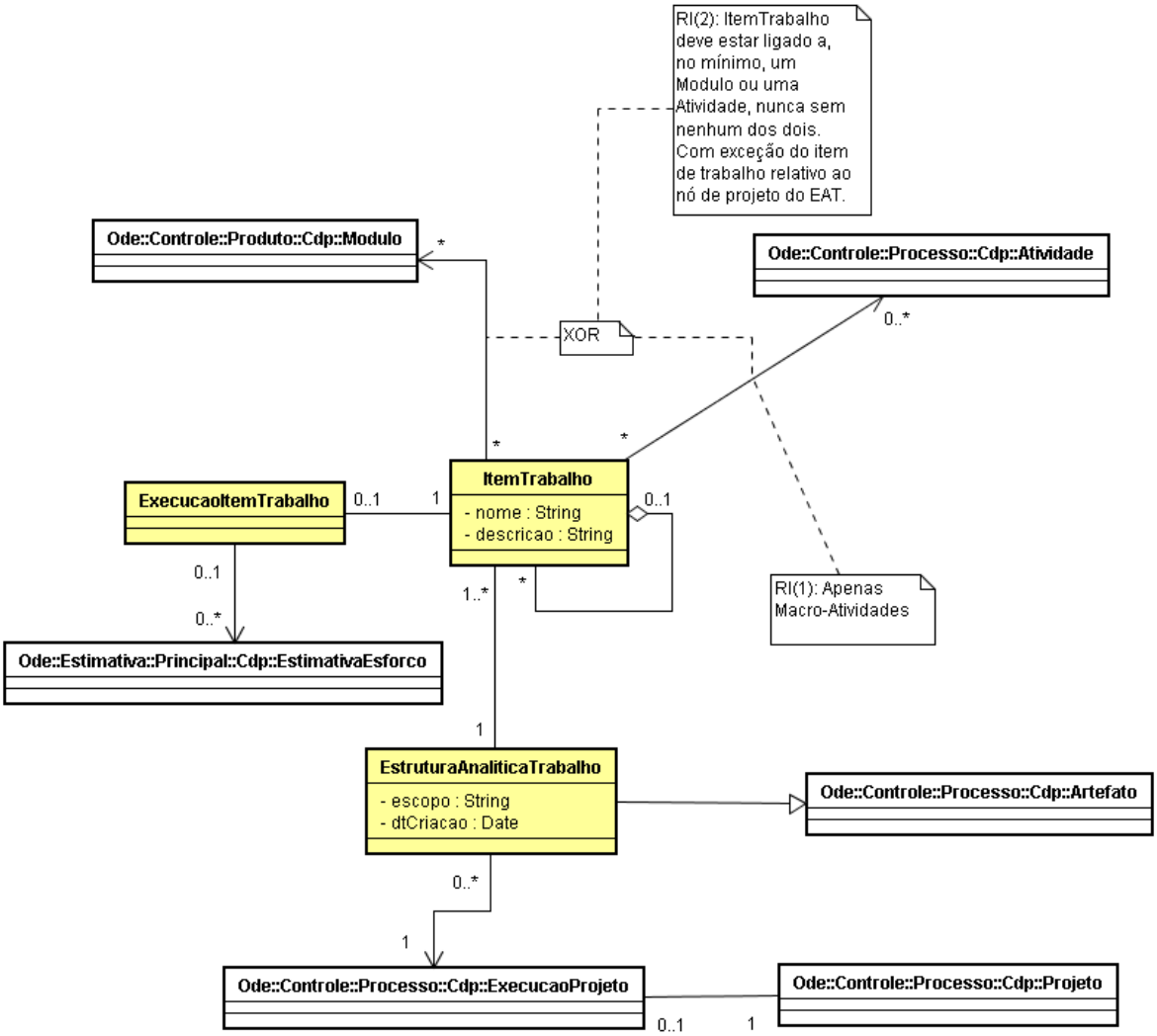

Figura 6.13 – Componente de Domínio do Problema do pacote *Eat*

A associação entre `ExecucaoItemTrabalho` e `Estimativa` (vide documento de análise) transforma-se, agora, em um relacionamento direto com `EstimativaEsforco`, que herda de Estimativa. Dessa forma podem ser associadas várias estimativas de esforço a um mesmo `ItemTrabalho`. Vale destacar que essa alteração decorre do fato de estarmos apenas tratando de estimativas de esforço nesta versão do trabalho.

6.5.2 – Componente de Gerência de Dados (Cgd)

As classes contempladas no Cgd têm a função de abstrair o acesso ao banco de dados usado na implementação do trabalho. A Figura 6.14 mostra o diagrama de classes desse pacote.

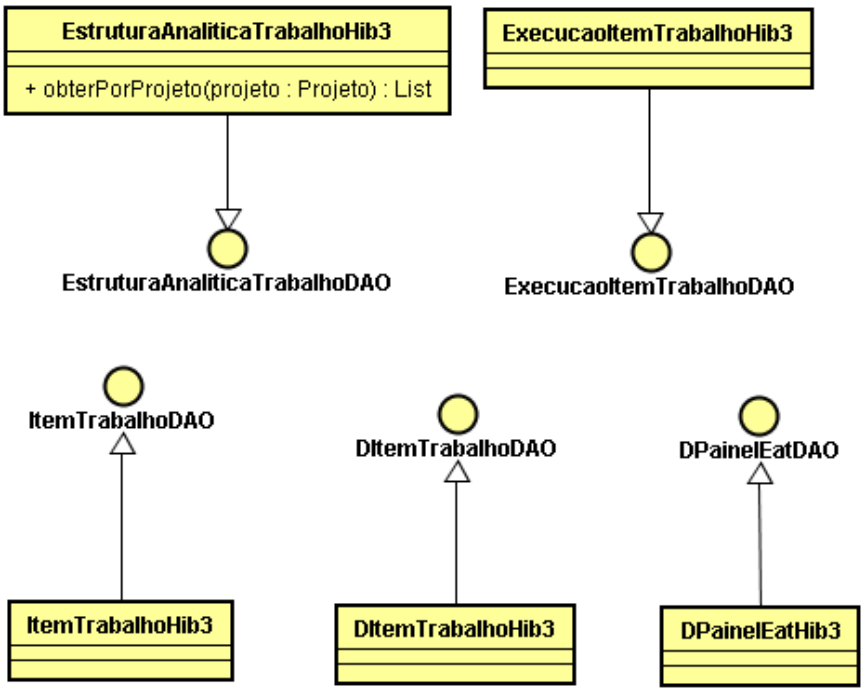

Figura 6.14 - Componente de Gerência de Dados do pacote *Eat*

Todas as classes mostradas na Figura 6.14 herdam da classe abstrata `Hib3DAO`, conforme discutido anteriormente.

6.5.3 – Componente de Gerência de Tarefas (Cgt)

O componente de Gerência de Tarefas desse pacote possui uma única classe, que é responsável por tratar todos os casos de uso definidos no documento de requisitos, como mostra a Figura 6.15.

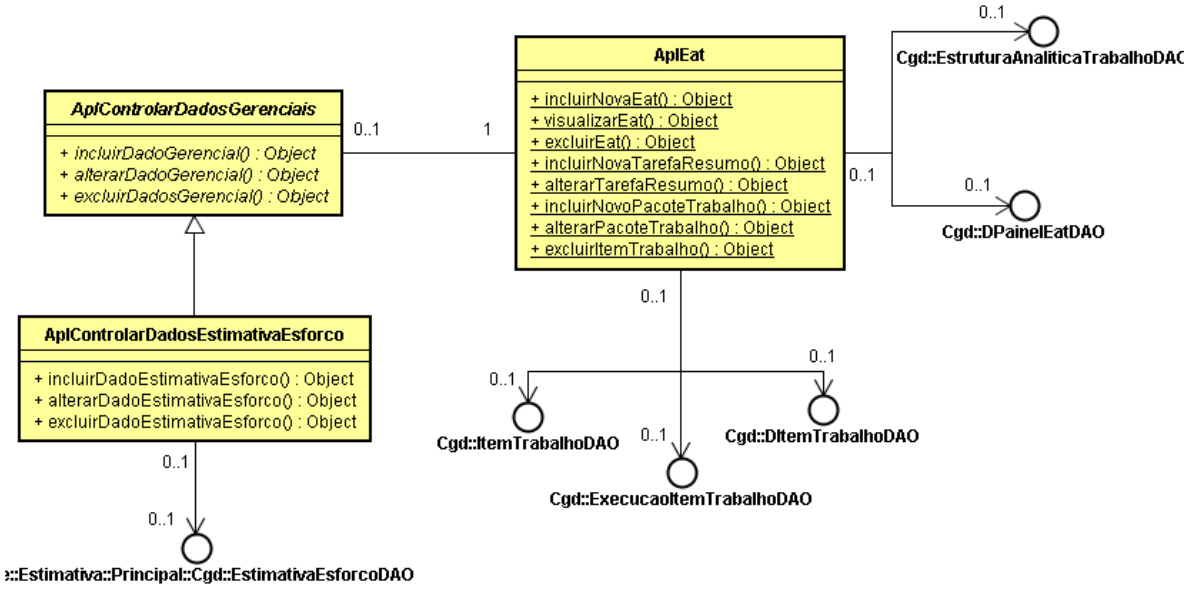

Figura 6.15 – Componente de Gerência de Tarefas do Pacote *Eat*

A classe abstrata `AplControlarDadosGerenciais` permite que sejam tratados vários tipos de dados gerenciais na ferramenta.

6.5.4 – Componente de Interação Humana (Cih)

O componente de Interação Humana do pacote *Eat* materializa as interfaces com o usuário para a realização dos casos de uso definidos no documento de especificação de requisitos. A Figura 6.16 mostra o diagrama de classes correspondente.

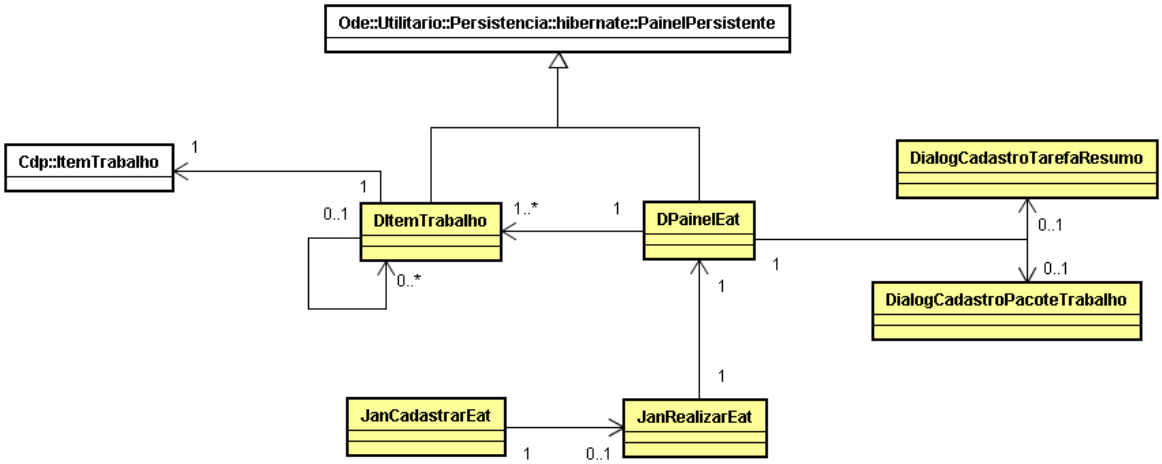

Figura 6.16 - Componente de Interação Humana do pacote *Eat*

JanCadastrarEat apresenta as EATs cadastradas no sistema e permite que se faça a criação de uma nova EAT, a alteração ou exclusão de uma EAT existente e a visualização de uma EAT. JanRealizarEat tem o objetivo de comportar o DPainelEat que contém todas as figuras representando a EAT como um todo. Além, por meio desse painel é possível solicitar o cadastro de tarefas-resumo e pacotes de trabalho, realizados pelas classes de interface DialogCadastroTarefaResumo e DialogCadastroPacoteTrabalho. Cada ItemTrabalho é representado graficamente por um objeto da classe DItemTrabalho.

6.5.5 – Componente de Controle de Interação (Cci)

As classes do pacote Cci são responsáveis por fazer o intercâmbio de dados entre as classes de aplicação e as classes de interface com o usuário da ferramenta, como mostra a Figura 6.17.

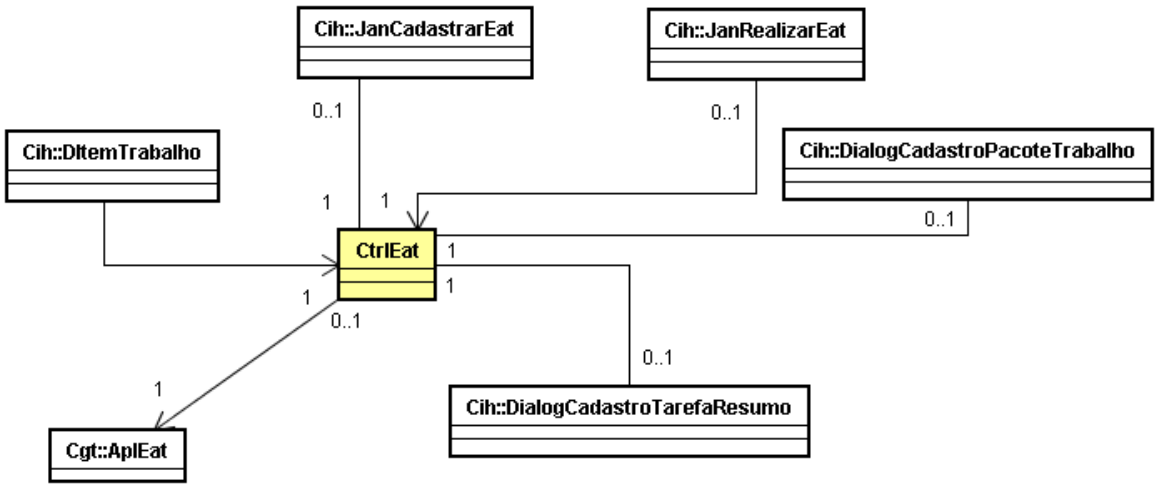

Figura 6.17 - Componente de Controle de Interação do Pacote *Eat*

Navegabilidade dupla entre JanRealizarEat e CtrlEat.

A navegabilidade no sentido CtrlEat e AplEat permite que as funcionalidades da ferramenta possam ser utilizadas por outras ferramentas. Além disso, se houver a necessidade de modificar a interface da ferramenta, basta reformular o controlador de acordo com mesma.

Toda vez que uma interface necessita acessar alguma funcionalidade, provida por uma classe de aplicação (AplEat), ela o faz via CtrlEat, que conhece as operações necessárias para atender à requisição da interface. Quando a aplicação requisitada retorna

um resultado para o controlador, esse interpreta o mesmo e atualiza as interfaces de acordo com a necessidade.

6.6 Implementação

Nesta seção são apresentadas algumas interfaces das ferramentas implementadas, a saber: Definição do Escopo (Figura 6.18) e Criação de um Novo Módulo (Figura 6.19) na ferramenta de Decomposição do Produto e Definição do Escopo de uma EAT (Figura 6.20) e Visualização de uma EAT (Figura 6.21) na ferramenta de apoio à elaboração de EATs.

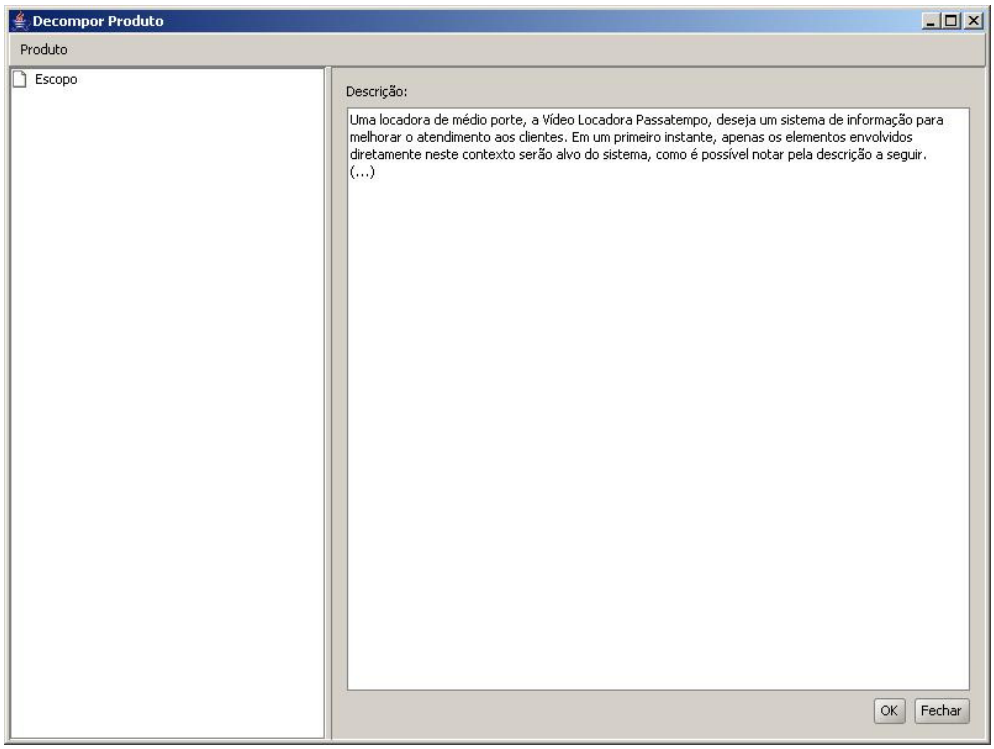

Figura 6.18 – Definição do Escopo.

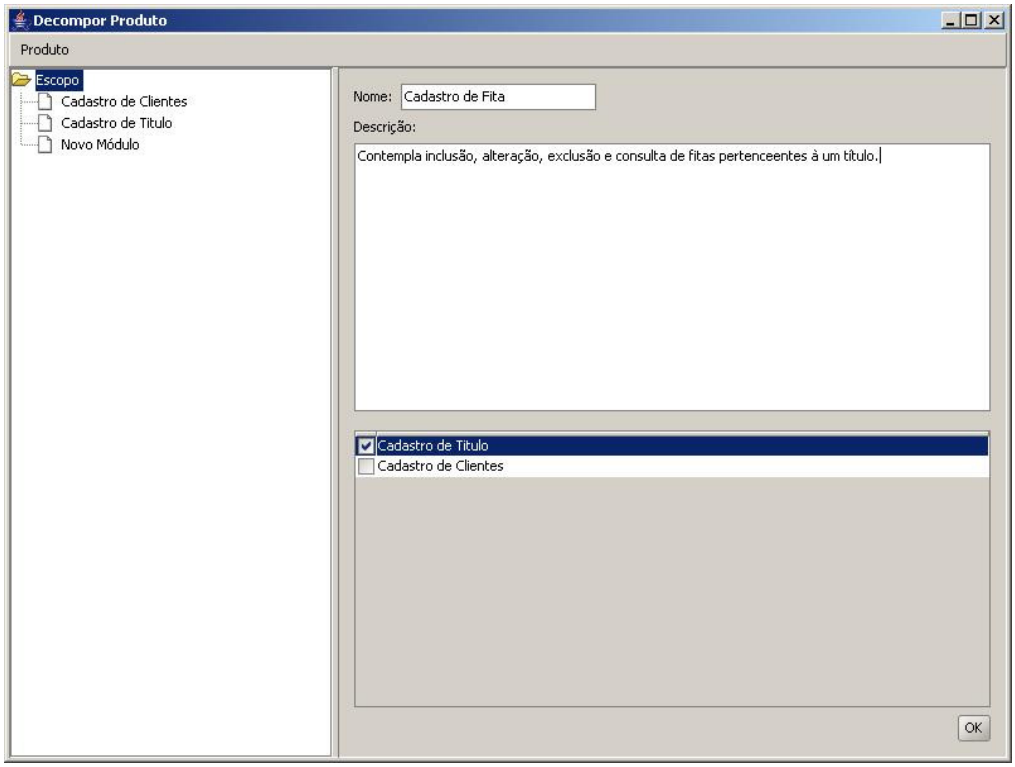

Figura 6.19 – Criação de um Novo Módulo.

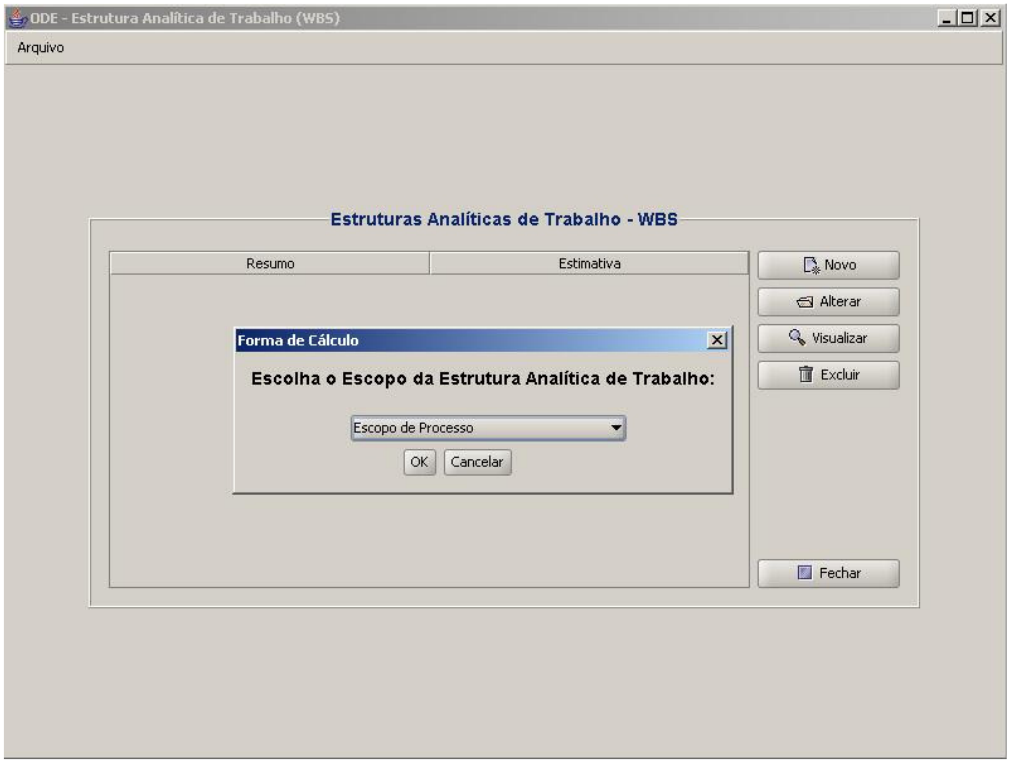

Figura 6.20 – Definição do Escopo de uma EAT

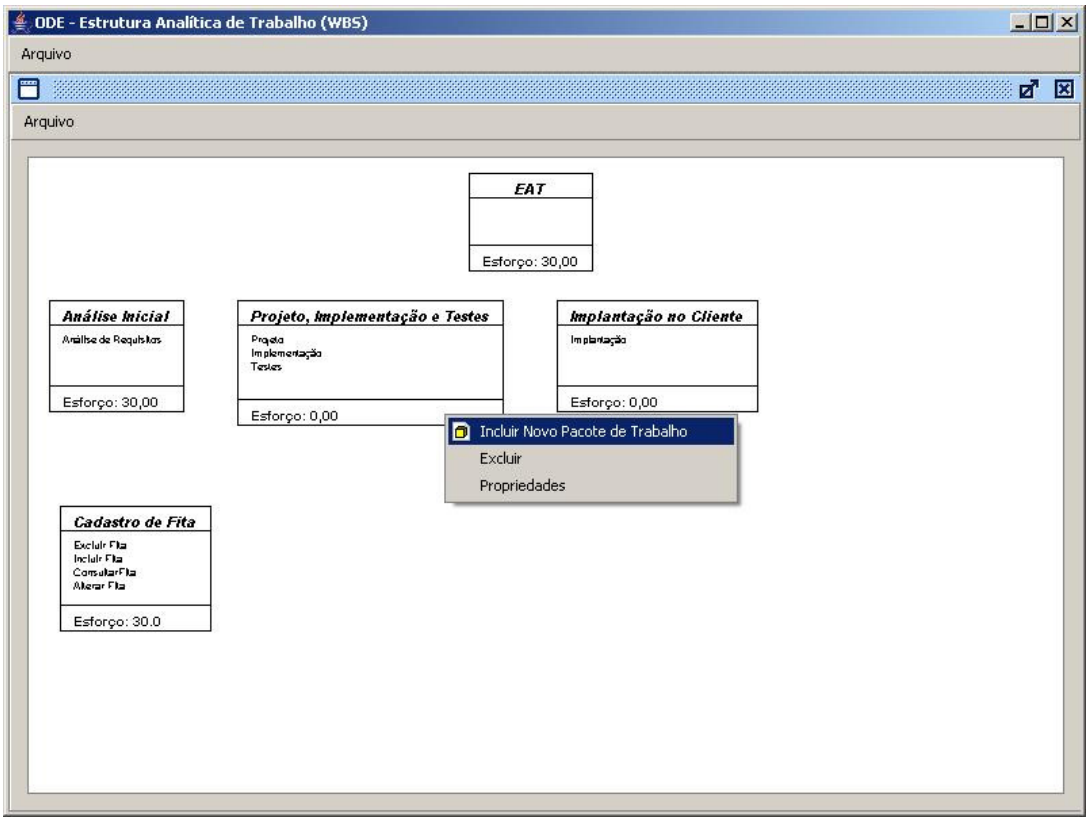

Figura 6.21 – Visualização de uma EAT

6.7 Testes

A abordagem de testes adotada neste trabalho foi baseada nos testes dos eventos de cada caso de uso implementado. Minimamente foi gerado um caso de teste para cada evento, normalmente seguindo diretamente a descrição do evento. Tome o caso de uso *Decompor Produto*, por exemplo. Foram criados casos de teste para os eventos de curso normal *Criar Novo Módulo*, *Alterar Dados de Módulo* e *Excluir Módulo*, e para os eventos de curso alternativo *Criar Novo Módulo* e *Alterar Dados de Módulo*.

Capítulo 7

Conclusões e Perspectivas Futuras

Neste capítulo são apresentadas as conclusões finais e perspectivas futuras referentes ao trabalho. A seção 7.1 disserta sobre as conclusões do trabalho. A seção 7.2 mostra algumas oportunidades de trabalhos futuros dentro do Projeto ODE, providas a partir do desenvolvimento deste trabalho.

7.1 Conclusões

A oferta de apoio automatizado às atividades relacionadas à gerência de projetos tende a aumentar a produtividade e até mesmo a qualidade dos produtos de software desenvolvidos em uma organização.

Neste trabalho foram desenvolvidas algumas funcionalidades de apoio à gerência de projetos, a saber:

- Ferramenta de Decomposição do Produto: visa analisar o projeto sob outro foco que não o de processo, decompondo-o e tratando suas partes gerenciáveis como módulos;
- Ferramenta de Construção de Estruturas Analíticas de Trabalho (EATs): tem como objetivo organizar tarefas de um projeto sob um foco de processo ou produto, integrando-se com as ferramentas de Decomposição do Produto e Definição de Processo. Além disso, a ferramenta provê uma visão macro sobre as estimativas de esforço sumarizadas de cada elemento de uma EAT;
- Ferramenta de Estimativa de Pontos de Função: apóia a elaboração de estimativas do tipo citado, tendo sido, na realidade, uma evolução da ferramenta proposta em (CRUZ, 2001), provendo, agora, três formas de cálculo diferentes, além de estar totalmente integrada à ferramenta de

Decomposição do Produto, permitindo o uso de módulos como itens de uma estimativa. Foi feito também um trabalho de re-estruturação da interface trazendo um novo padrão de interface que visa a facilitar o uso de ferramentas internas ao ambiente ODE;

- Ferramenta de Estimativa de Pontos de Caso de Uso: tem por objetivo apoiar a realização de estimativas do tipo citado, tendo sido, na realidade, uma evolução da ferramenta proposta em (LAHAS, 2005), integrando-se, agora, com a ferramenta de modelagem UML do ambiente ODE (OODE) (CARREIRA, 2003). Além disso, a re-estruturação dessa ferramenta permitiu que fossem feitas várias estimativas para um mesmo projeto usando o mesmo padrão de interface da ferramenta de Estimativa de Pontos de Função.

Este trabalho permitiu o aprendizado de diversas tecnologias e metodologias, dentre eles: (i) uso de camadas de persistências; (ii) padrões de projeto; (iii) criação de gráficos 2D em Java; (iv) integração de ferramentas; (v) geração de relatórios em Java; (vi) orientação a objetos; e (vii) gerenciamento de projetos.

Além disso, foi uma oportunidade para aplicar os conhecimentos adquiridos durante o curso de Ciência da Computação com a ênfase em Sistemas de Informação.

7.2 Perspectivas Futuras

É muito comum, e muitas vezes necessário, que a finalização de um trabalho acarrete na construção ou manutenção de outros e faça com que um sistema ou grupo de sistemas cresça sempre no sentido de melhor atender às requisições dos clientes.

Dando continuidade ao trabalho, é necessário que sejam implementados todos os eventos de casos de uso propostos que não foram contemplados na fase de projeto e implementação deste trabalho, a saber:

- A Decomposição do Produto deve permitir a caracterização de módulos. Essa caracterização pode servir como apoio no momento de uma busca por módulos similares;

- Na ferramenta de Estimativas de Pontos de Função, deve-se permitir a inclusão de arquivos de dados de um projeto em outro, atendendo ao cenário *Incluir Arquivo de Dados no Projeto Corrente* do caso de uso *Cadastrar Arquivos* (encontrado no anexo A, subseção A.1.2);
- Garantir que a seleção de um caso de uso na ferramenta de Estimativas de Pontos de Caso de Uso, acarrete na seleção automática dos atores relacionados ao caso de uso;
- A ferramenta de Criação de Estruturas Analíticas de Trabalho deve permitir a persistência dos painéis que contém as EATs.

Além da finalização das funcionalidades citadas, o trabalho abre caminho para vários outros projetos, tais como:

- Desenvolvimento de outras ferramentas de apoio à realização de estimativas, sobretudo tempo e custo;
- Manipulação de outros dados gerenciais pela ferramenta de apoio à elaboração de EATs, como estimativas de tempo e custo, alocação de recursos etc;
- Criação de uma ferramenta de cronogramação que gere cronogramas com base nas EATs de um projeto.

Referências Bibliográficas

ANDRADE, E.L.P., **Pontos de Casos de Uso e Pontos de Função na gestão de estimativa de tamanho de projetos de software orientados a objetos**. Dissertação de Mestrado, Programa de Pós-Graduação em Gestão de Conhecimento e Tecnologia da Informação, Universidade Católica de Brasília. Brasília, 2004.

BAUER,C., KING, G., *Hibernate em Ação*, Editora Ciência Moderna, 1ª edição, 2005.

BERTOLLO, G.; RUY, F.B.; MIAN, P.G.; PEZZIN, J.; SHWAMBACH, M.M.; NATALI, A.C.C.; FALBO, R.A., "ODE: Um ambiente de Desenvolvimento de Software Baseado em Ontologias". Anais do XVI Simpósio Brasileiro de Engenharia de Software - SBES'2002. Caderno de Ferramentas, pp.438-443, Gramado - RS, Brasil, Outubro 2002.

BOOCH, G., RUMBAUGH, J, JACOBSON, I., *UML Guia do Usuário*. 2ª edição, Editora Campus, 2005.

CARVALHO, V.A.; ARANTES, L.O.; FALBO, R.A., "EstimaODE: Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento de Software ODE". V Simpósio Brasileiro de Qualidade de Software, SBQS'2006, pp 12-26. Vitória, Espírito Santo, Maio 2006.

CRUZ, A. P. **Uma Ferramenta CASE para Análise de Pontos de Função,** Projeto de Graduação, Curso de Ciência da Computação, Universidade Federal do Espírito Santo, 2001.

FALBO, R. A. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software**. Tese de Doutorado, COPPE/UFRJ, RJ, Dezembro, 1998.

FALBO, R.A.; RUY, F.B.; MORO, R.D., "Using Ontologies to Add Semantics to a Software Engineering Environment". 17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2005, p. 151 - 156, Taipei, China, July 2005.

FALBO, R.A., RUY, F.B., PEZZIN, J., DAL MORO R., "Ontologias e Ambientes de Desenvolvimento de Software Semânticos". Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería Del Conocimiento, JIISIC'2004, Volumen I, pp. 277-292, Madrid, España, Noviembre 2004.

FALBO, R.A., **Engenharia de Software: Notas de Aula 2005,** Documento Eletrônico disponível em <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-2/NotasDeAula.pdf >. Acesso em 27/06/2006.

GAMMA, E., HELM R., JOHNSON R., VLISSIDES, J., *Design Patterns - Elements of Reusable Object-oriented Software***.** Addison-Wesley Professional Computing Series, 1995.

LAHAS, L., **Uma Ferramenta de Apoio à Realização de Estimativas de Ponto de Caso de Uso,** Projeto de Graduação, Curso de Ciência da Computação, Universidade Federal do Espírito Santo, 2005.

MARTINS, J.C.C., *Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML*. 2ª edição. Rio de Janeiro: Brasport, 2005.

MEYER, B., *Object-Oriented Software Construction*, Second Edition, Prentice Hall, 1997.

PMI, **PMBOK: Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos**. 3ª edição. 2004.

PRESSMAN, R. S., *Engenharia de Software*. 5ª edição. Rio de Janeiro: McGrawHill, 2002.

RUY, F.B., **Infra-estruturas de Apoio à Integração de Dados e Conhecimento em ODE**, Projeto de Graduação, Curso de Ciência da Computação, Universidade Federal do Espírito Santo, 2003.

SCHNEIDER et al., *Applying Use Cases: A Pratical Guide.* Addison-Wesley Longman, Inc, 2001.

SUN Developer Network, "*Data Access Object*", 2001. Disponível em: <java.sun.com/j2ee/patterns/DataAccessObject.html >. Acesso em: 11 jul. 2006.

VAZQUEZ, C. E.; SIMOES, G. S.; ALBERT, R. .M., *Análise de Pontos de Função: medição, estimativas e gerenciamento de projetos de software.* 3ª edição, São Paulo: Editora Érica, 2005.

Anexo A

Documentação da Evolução da Ferramenta de Apoio a Estimativas Usando Pontos de Função

Este documento tem o intuito de prover informações sobre a versão atual da ferramenta de Estimativas de Pontos de Função, abrangendo todos as fases do ciclo de vida de desenvolvimento.

Inicialmente é apresentada a especificação de requisitos da ferramenta (seção A.1), detalhando apenas os requisitos funcionais, já que os requisitos não-funcionais foram tratados na seção 4.3. Em seguida, na seção A.2, é discutida a fase de análise da ferramenta, apresentando uma visão do domínio do problema sem levar em conta qualquer aspecto tecnológico. Finalmente a seção A.3 trata do projeto dessa versão da ferramenta.

A.1 Especificação de Requisitos

Nesta seção é apresentada a Especificação de Requisitos da ferramenta de Estimativa de Pontos de Caso de Uso, iniciando com uma descrição do problema (seção A.1.1) e em seguida apresentando a modelagem dos casos de uso levados em conta e as descrições dos mesmos (seção A.1.2).

A.1.1 Descrição do Mini-Mundo

A Análise de Pontos de Função (APF) é um método surgido no início da década de 1970 no parque de pesquisas da IBM. Foi inicialmente usada para medir a produtividade da equipe num ambiente que havia grande número de projetos e grande número de linguagens de programação usadas (VAZQUEZ et. al, 2005). Posteriormente foi criado um órgão internacional, o IFPUG – *International Function Point Users Group,* com o intuito de pesquisar o método e padronizá-lo, devido à crescente necessidade das organizações estimarem esforço, tempo e custos de seus projetos.

A primeira versão da ferramenta de Estimativas de Ponto de Função em ODE foi desenvolvida em (CRUZ, 2001). Até então a ferramenta permitia a contagem de PFs para projetos de desenvolvimento sob um foco do projeto por completo, não sendo possível elaborar contagens para subsistemas ou funcionalidades específicas de um projeto. Além disso, a ferramenta contava apenas com a forma de cálculo estabelecida pelo IFPUG. Viu-se, então, a necessidade de reestruturação da ferramenta com o intuito de permitir contagens para partes menores do projeto, além de possibilitar a contagem segundo abordagem de Contagem Estimativa da NESMA.

A Figura A.1 mostra o diagrama de pacotes da ferramenta. No pacote *Principal* estão os elementos gerais utilizados na realização de estimativas em ODE, tanto sob o contexto de escopo (projeto, módulo etc) quanto de natureza da estimativa (tamanho, esforço, custo e tempo). Já o subsistema *PontoFuncao* diz respeito especificamente ao apoio à Análise por Pontos de Função. Esse subsistema possui dois pacotes: o pacote *Analise*, que trata das funcionalidades ligadas diretamente à contagem de PFs, e o pacote *BaseCalculo*, que se refere às funcionalidades de apoio para registrar informações básicas do método, tais como funções, tabelas de complexidade para as funções etc. Neste

documento é discutido apenas o pacote *Analise*, já que não houve mudanças significativas nos modelos do pacote *BaseCalculo* (CRUZ, 2001).

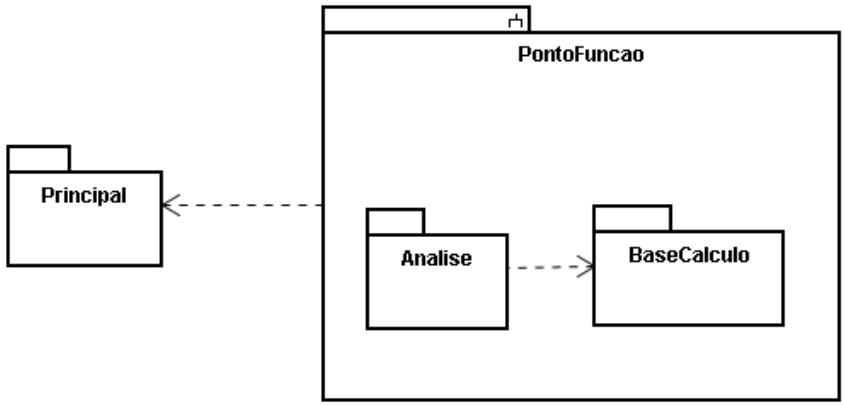

Figura A.9 - Diagrama de Pacotes

.

A.1.2 Modelo de Casos de Uso

A Figura A.2 mostra o diagrama de casos de uso do pacote *Analise*. Os casos de uso mostrados nessa figura são provenientes da re-estruturação da especificação proposta em (CRUZ, 2001).

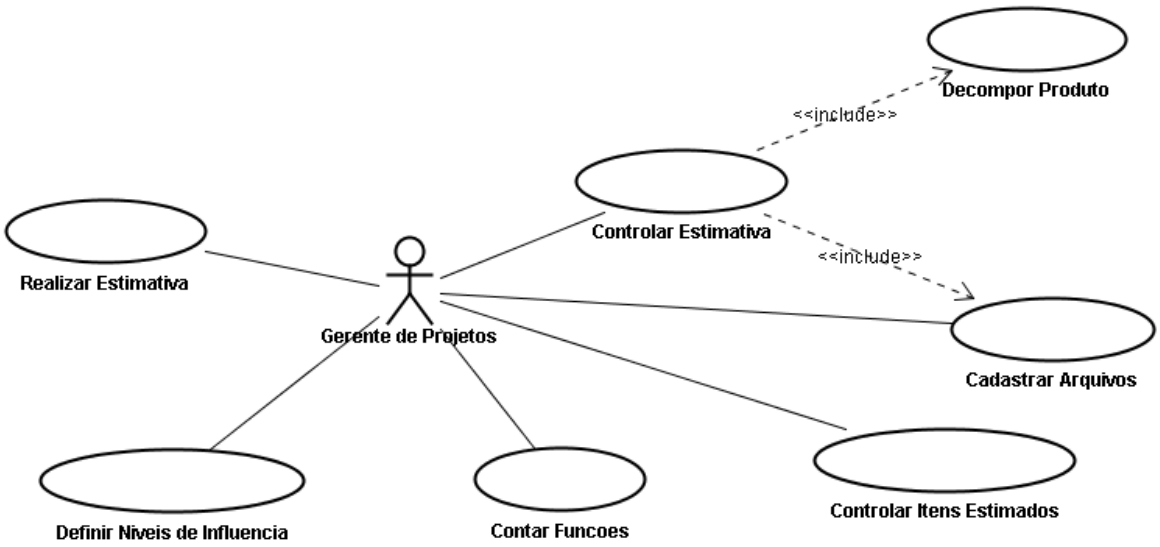

Figura A.2 - Diagrama de Casos de Uso do pacote Analise

Ao se iniciar uma estimativa o escopo de contagem é definido para o projeto como um todo ou para uma seleção de módulos do mesmo. Ao se escolher o escopo de

seleção de módulos, é interessante permitir que o usuário refine a decomposição do produto, podendo incluir, alterar ou excluir módulos do projeto. Da mesma forma, depois de definido o escopo de contagem, é interessante que os arquivos (funções de dados) da contagem possam ser cadastrados no sistema.

A seguir, os casos de uso propostos são descritos.

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Controlar Estimativa

Descrição: Este caso de uso permite criar uma nova estimativa por ponto de função para o projeto corrente, abrir, copiar ou excluir uma estimativa por ponto de função existente do projeto corrente.

Curso Normal:

Criar Nova Estimativa

O gerente informa a forma de cálculo a ser usada na estimativa por pontos de função: Contagem Estimativa da NESMA, Contagem Padrão IFPGU com Pontos de Função Não Ajustados ou Contagem Padrão IFPGU Completa, com Pontos de Função Ajustados. Em ODE, o projeto corrente pode ser ou um projeto de, desenvolvimento ou um projeto de manutenção. O tipo de contagem de PF a ser aplicado será relativo ao tipo do projeto. Ou seja, se o projeto corrente for um projeto de desenvolvimento, o tipo da contagem será contagem de PF de projeto de desenvolvimento, caso contrário, será uma contagem de PFs de projeto de manutenção. Adicionalmente o gerente informa se o escopo da contagem será o projeto como um todo ou apenas algumas de suas funções. Caso o escopo seja de algumas funções do projeto, permite-se refinar a decomposição do produto, realizando o caso de uso "Decompor Produto", bem como definir arquivos de dados do projeto, realizando o caso de uso "Cadastrar Arquivos".

Alterar Estimativa

O gerente informa a estimativa ainda não finalizada que deseja alterar. Os dados da estimativa são exibidos e o gerente pode editar a mesma.

Copiar Estimativa:

O gerente informa a estimativa finalizada que deseja copiar. É exibida uma mensagem pedindo que seja informada a forma de cálculo que será utilizada. Um nova estimativa é criada com os dados da estimativa selecionada e com a forma de cálculo informada.

Excluir Estimativa:

O gerente informa a estimativa que deseja excluir. Uma confirmação é requisitada e, caso confirmada, a estimativa é excluída.

Visualizar Relatório de Estimativa

O gerente informa a estimativa que deseja visualizar um relatório. São exibidos os dados daquela estimativa em forma de relatório.

Finalizar Estimativa

O gerente informa a estimativa que deseja finalizar. Uma mensagem de confirmação é exibida, mostrando que uma estimativa finalizada não poderá mais ser alterada. Após a confirmação, a data de finalização da estimativa é registrada.

Curso Alternativo:

Alterar Estimativa:

A estimativa selecionada está finalizada: uma mensagem é exibida informando que não é possível alterar estimativas finalizadas.

Classes: EstimativaPF, Projeto, ItemEstimado, Funcao, ValorMedida, ArquivoDados, NivelInfluencia.

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Controlar Itens Estimados

Descrição: Este caso de uso permite selecionar e classificar itens a serem contados em uma estimativa (módulos e arquivos de dados).

Curso Normal:

Selecionar Módulos:

Pré-condição: O escopo da contagem não pode ser o projeto como um todo.

São exibidos todos os módulos do projeto (resultado da decomposição do produto de software). O gerente de projeto seleciona os módulos que deseja incluir na contagem e os mesmos são tratados como funções transacionais a partir deste ponto.

Selecionar Arquivo de Dados:

Pré-condição: O escopo da contagem não pode ser o projeto como um todo.

São exibidos todos os arquivos de dados do projeto registrados. O gerente de projeto seleciona os arquivos que deseja incluir na contagem e os mesmos são tratados como funções de dados a partir deste ponto.

Classificar Módulos:

São apresentados os módulos que fazem parte do escopo da contagem. O gerente classifica cada módulo segundo seu tipo de função transacional, a saber: Entrada Externa (EE), Consulta Externa (CE) e Saída Externa (SE). A classificação é registrada para cada item.

Classificar Arquivo de Dados:

São apresentados os arquivos de dados que fazem parte do escopo da contagem. O gerente classifica cada arquivo de dados segundo seu tipo de função de dados, a saber: Arquivo Lógico Interno (ALI) e Arquivo de Interface Externa (AIE). A classificação é registrada para cada item.

Classes: EstimativaPF, ItemEstimado, ArquivoDados, ExecucaoModulo

Restrição de Integridade: Apenas módulos do projeto que está sendo estimado podem ser selecionados como itens a serem estimados na estimativa.

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Cadastrar Arquivos

Descrição: Este caso de uso permite criar, excluir, alterar e consultar arquivos de dados de um projeto.

Curso Normal:

Criar Novo Arquivo de Dados:

O gerente informa o nome e descrição do novo arquivo de dados. Os dados são validados e um novo arquivo de dados é criado associado ao projeto corrente.

Incluir Arquivos de Dados no Projeto Corrente

O gerente informa os arquivos de dados que deseja incluir no projeto corrente e os arquivos passam a estar no escopo do projeto corrente.

Alterar Arquivo de Dados:

O gerente informa o arquivo de dados que deseja alterar e os novos dados. Os dados são validados e registrados.

Consultar Arquivo de Dados:

O gerente informa o arquivo de dados que deseja consultar. Os dados desse arquivo são apresentados.

Excluir Arquivo de Dados:

O gerente informa o arquivo de dados que deseja excluir. Uma solicitação de confirmação é exibida e, caso confirmada, o arquivo de dados é excluído. Só é permitida a exclusão de arquivos de dados que não estejam relacionados com nenhuma contagem.

Curso Alternativo:

Excluir Arquivo de Dados:

O arquivo de dados está sendo utilizado em alguma contagem: Uma mensagem é exibida, informando que o arquivo de dados em questão não poderá ser excluído, pois existem uma ou mais contagens relacionadas a ele.

Classes: ArquivoDados, Projeto

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Contar Funções

Descrição: Este caso de uso trata da contagem das funções de dados e transacionais que fazem parte do escopo da estimativa.

Pré-condição: A forma de contagem selecionada para a estimativa tem de ser a padrão IFPUG ajustada ou não ajustada.

Curso Normal:

Contar Funções de Dados:

O gerente seleciona uma função de dados (AIE ou ALI) e informa os números de registros lógicos e de itens de dados. A complexidade da função é computada e os dados são registrados.

Contar Funções Transacionais:

O gerente seleciona uma função transacional (EE, CE ou SE) e informa os números de arquivos e de itens de dados referenciados. A complexidade da função é computada e os dados são registrados.

Classes: EstimativaPF, ItemEstimado, Funcao .

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Definir Níveis de Influência

Descrição: Este caso de uso permite definir os níveis de influência para uma contagem em uma estimativa de pontos de função utilizando a forma de cálculo padrão IFPUG com pontos ajustados.

Pré-condição: Forma de cálculo da contagem deve ser IFPUG com pontos ajustados.

Curso Normal:

O gerente define um nível de influência, um inteiro entre 0 e 5, para cada uma das 14 características gerais do sistema, a saber:

1. Comunicação de Dados
2. Processamento de Dados Distribuído
3. Performance
4. Utilização do Equipamento
5. Volume de Transações
6. Entrada de Dados *On-line*
7. Eficiência do Usuário Final
8. Atualização *On-line*
9. Processamento Complexo
10. Reusabilidade
11. Facilidade de Implantação
12. Facilidade Operacional
13. Múltiplos Locais
14. Facilidade de Mudanças

Caso os dados sejam válidos, os níveis de influência são registrados.

Curso Alternativo:

Algum nível de influência não está entre 0 e 5: uma mensagem é mostrada informando que os níveis de influência devem ser inteiros entre 0 e 5.

Classes: EstimativaPF, NivelInfluencia, CaracteristicaGeral

Sub-Sistema: *PontoFuncao* (pacote *Analise*)

Caso de Uso: Realizar Estimativa

Descrição: Este caso de uso calcula os pontos de função da estimativa.

Curso Normal:

Calcular PFs não ajustados segundo a Contagem de Estimativa da NESMA:

São obtidos os números de itens estimados para cada tipo de função (ALI, AIE, EE, SE e CE) e o valor de pontos de função é calculado segundo a seguinte fórmula:

Total PF = 7*numItensALI + 5*numItensAIE + 4*numItensEE+ 5*numItensSE + 4*numItensCE

onde numItens<Função> corresponde ao número de itens dessa contagem que pertencem à função <Função>.

Os dados da contagem são exibidos.

Calcular PFs não ajustados segundo a Contagem Padrão IFPGU:

Pré-condição: A forma de cálculo da estimativa corrente tem de ser Contagem Padrão IFPGU (com Pontos de Função Ajustados ou não).

Para cada um dos cinco tipos de função existente (ALI, AIE, EE, SE e CE) são computados os totais de pontos de função (NPF_i) segundo a seguinte expressão:

$$NPF_i = \sum NC_{i,j} * C_{i,j}, \text{ com } 1 \leq j \leq 3$$

onde:

$NC_{i,j}$ = número de itens de um único tipo de função que foram classificados na complexidade j.

$C_{i,j}$ = valor da contribuição da complexidade j no cálculo dos pontos da função i, sendo que j varia de 1 a 3, segundo os valores de complexidade (Complexa, média, Simples) e i varia de 1 a 5, segundo os tipos de função existentes (ALI, AIE, EE, SE e CE).

O total de pontos de função não ajustados (PFNA) é dado pelo somatório dos pontos das tabelas de função:

$$PFNA = \sum NPF_i \text{ , com } 1 \leq i \leq 5 \text{ (segundo os tipos de função existentes).}$$

Os dados da contagem são exibidos.

Calcular PFs ajustados segundo a Contagem Padrão IFPGU:

Pré-condição: A forma de cálculo da estimativa corrente tem de ser Contagem Padrão IFPGU Completa, com Pontos de Função Ajustados e os níveis de influência das características gerais têm de ter sido previamente definidos.

Primeiramente realiza-se o evento de caso de uso "*Calcular PFs não ajustados segundo a Contagem Padrão IFPGU*". A seguir, é calculado o nível de influência total (NIT), com base nos níveis de influência (NI) previamente definidos (vide caso de uso "Definir Níveis de Influência para Cálculo do Fator de Ajuste"):

$$NIT = \sum NI_i \text{ , com } 1 \leq i \leq 14$$

A seguir é calculado o valor do fator de ajuste (VFA), dado pela fórmula:

$$VFA = (NIT*0,01)+0,65$$

Finalmente calcula-se o total de pontos de função ajustados (PFA) com a seguinte fórmula:

$$PFA = PFNA*VFA$$

Os dados da contagem são exibidos.

Classes: EstimativaPF, ItemEstimado, Funcao, ValorMedida, Medida, NivelInfluencia, Contribuicao.

A.2 Análise

Nesta seção é apresentada a Especificação de Análise do projeto de manutenção da ferramenta de Estimativas de Pontos de Função.

Na seção A.2.1, é apresentado um diagrama de pacotes mostrando o relacionamento entre os pacotes do ambiente ODE e os pacotes específicos da ferramenta-alvo. Em seguida, a seção A.2.2 apresenta e discute os diagramas de classes derivados a partir da especificação de requisitos.

A.2.1 Modelo de Classes

O diagrama de pacotes da Figura A.3 mostra as dependências existentes entre os pacotes contemplados nessa ferramenta.

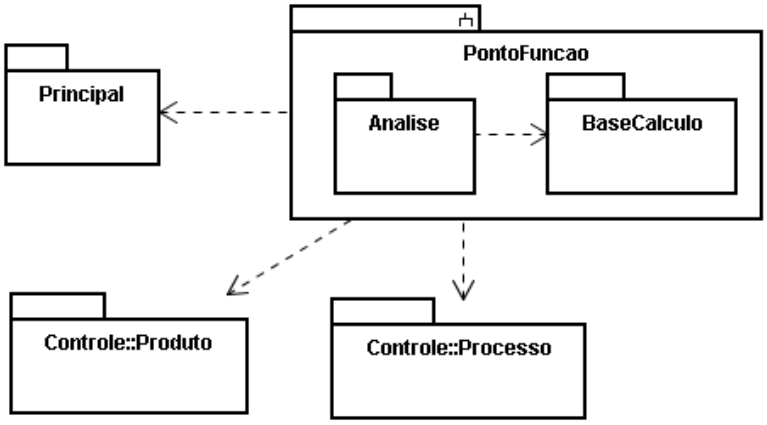

Figura A.10 - Diagrama de pacotes

O pacote *Principal* contém a infra-estrutura utilizada nas ferramentas de estimativa de ODE. O pacote *PontoFuncao* possui as classes que materializam a ferramenta de Estimativa de Pontos de Função no ambiente. A seguir são apresentados os diagramas de classe da fase de análise.

A.2.2 Diagramas de Classes

A seguir são apresentados os diagramas de classe dos pacotes *Analise* e *BaseCalculo*. É válido lembrar que o trabalho de reestruturação foi feito com o foco sobre o pacote *Analise*.

A.2.2.1 – Pacote Analise

O modelo da Figura A.4 mostra uma infra-estrutura em que diferentes tipos de estimativas, organizados pela natureza da estimativa sendo realizada (atualmente, de esforço e tamanho), ou pelo escopo da estimativa (atualmente, para o projeto como um todo ou para um conjunto de módulos do projeto), são abrigados como especializações da classe `Estimativa`. As classes `ExecucaoProjeto` e `ExecucaoModulo` isolam informações sobre a execução e as estimativas das respectivas classes de controle de ODE, `Projeto e Modulo`. Isso é importante, uma vez que essas classes são muito utilizadas por diversas ferramentas do ambiente e, portanto, idealmente, devem ser mantidas o mais simples possível.

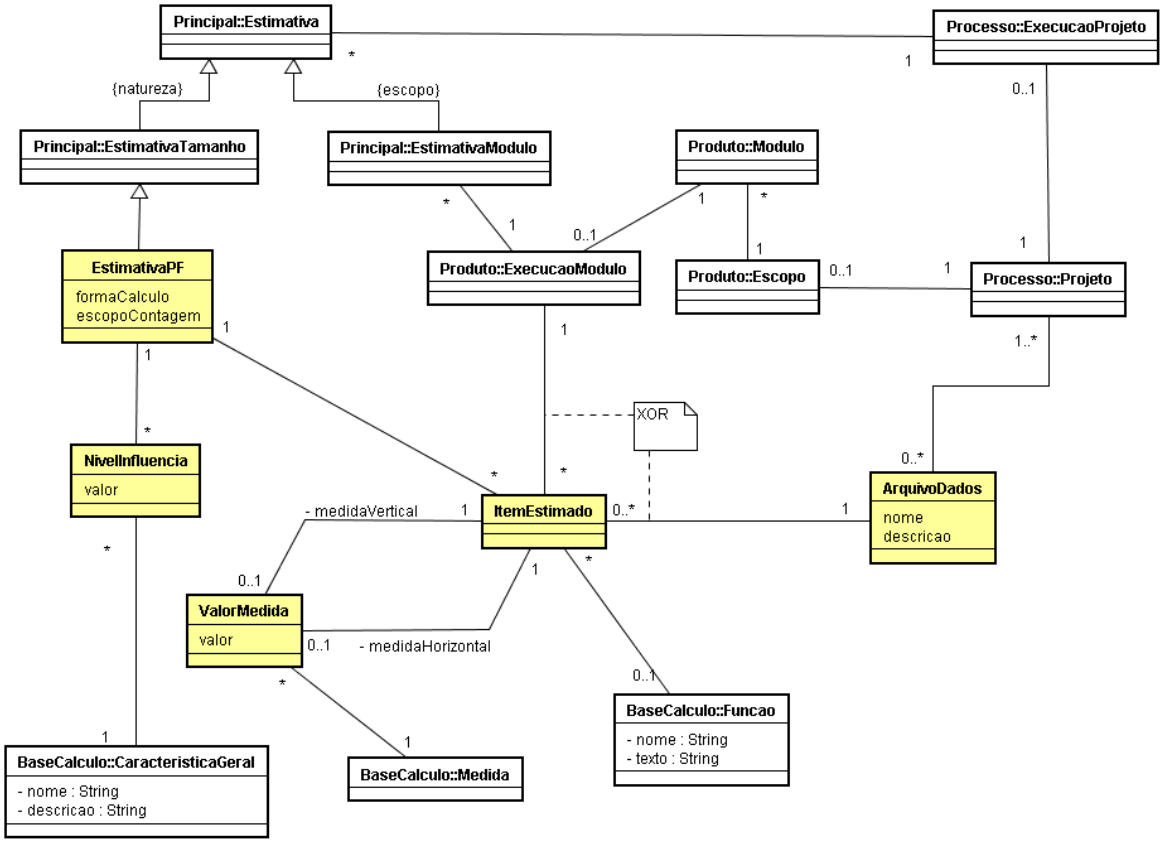

Figura A.11 - Diagrama de Classes do pacote Analise

A subclasse `EstimativaPF` representa estimativas de tamanho realizadas usando o método da Análise de Pontos de função. A forma de cálculo e o escopo de contagem escolhidos são registrados nos objetos dessa classe. Uma estimativa de Pontos de Função está relacionada diretamente com os itens estimados na contagem (objeto da classe

`ItemEstimado`). Em um item estimado são registradas as seguintes informações: a que tipo de função o item se refere (arquivo lógico interno, arquivo de interface externa, entrada externa, saída externa ou consulta externa); a qual funcionalidade (módulo ou arquivo de dados) o item se refere (`ExecucaoModulo` ou `ArquivoDados`); e quais os valores de medida horizontal e vertical (arquivos referenciados, registros lógicos ou itens de dados registrados pela classe `Medida`). Se a forma de contagem utilizada for a padrão IFPUG com pontos de função ajustados, os níveis de influência das 14 características gerais deve ser registrado (objetos da classe `NivelInfluencia` que se relaciona diretamente com a classe `CaracteristicaGeral`).

A.2.2.2 – Pacote BaseCalculo

A análise de pontos de função considera cinco tipos de função: Arquivo Lógico Interno (ALI), Arquivo de Interface Externa (AIE), Entrada Externa (EE), Saída Externa (SE) e Consulta Externa (CE). Cada um desses tipos é representado pela classe Função na Figura A.5.

Cada função possui uma tabela que relaciona as faixas de valores (limites) de suas duas medidas para determinar a complexidade (simples, média ou complexa, instâncias da classe `Complexidade`) de uma funcionalidade dessa função. A identificação da complexidade é feita enquadrando cada uma das duas medidas em uma faixa de valores e definindo a complexidade. As classes *Medida*, *Limite* e *IdentificacaoComplexidade* modelam essas tabelas, como ilustra a Figura A.6.

A classe CaracteristicaGeral descreve as catorze características gerais definidas pela abordagem IFPUG com pontos de função ajustados.

Por fim, a classe *Contribuicao* registra o peso de cada complexidade (simples, média ou complexa) no cálculo dos pontos de uma dada função.

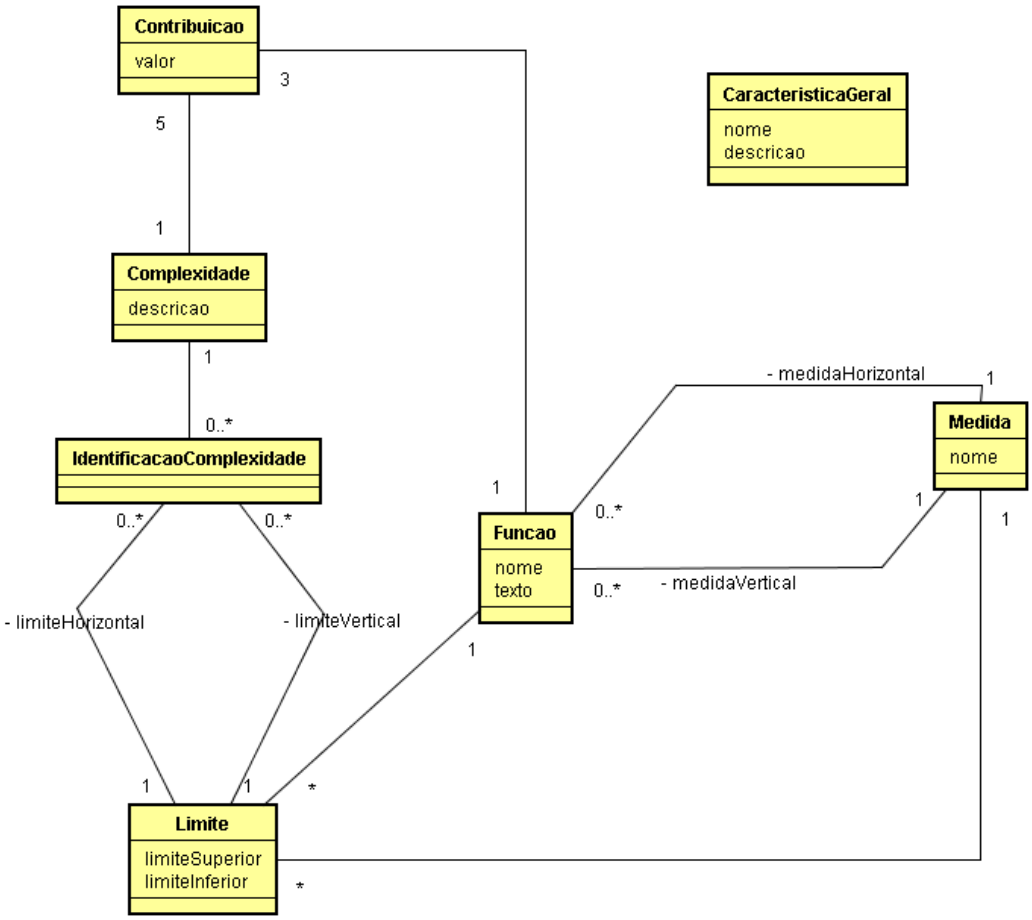

Figura A.12 - Diagrama de Classes do pacote BaseCalculo

Medidas

Limites

Identificação de Complexidade

Número de Arquivos Referenciados	Itens de Dados Referenciados		
	De 1 a 4	De 5 a 15	16 ou mais
0 ou 1	Simples	Simples	Média
2	Simples	Média	Complexa
3 ou mais	Média	Complexa	Complexa

Figura A.6 – Tabela de Identificação da Complexidade das Entradas Externas (WEBER, 2001).

A.3 Projeto e Implementação

Nesta seção é apresentada a Especificação de Projeto do projeto de manutenção da ferramenta de Estimativas de Pontos de Função.

Nesse ponto do projeto todos os resultados obtidos na fase de análise são expostos aos aspectos tecnológicos adotados no ambiente ODE, tais como a linguagem de programação, mecanismo de persistência adotado e arquitetura de componentes adotada.

A seção A.3.1 apresenta a organização física dos pacotes que compõem a ferramenta. Em seguida as seções A.3.2 e A.3.3 apresentam e discutem os diagramas de classes dos pacotes encontrados na ferramenta para cada componente definido na arquitetura. Finalmente a seção A.3.4 apresenta algumas interfaces da ferramenta.

A.3.1 Organização dos Pacotes

O diagrama de pacotes da Figura A.7 mostra as dependências existentes entre os pacotes físicos contemplados nessa ferramenta.

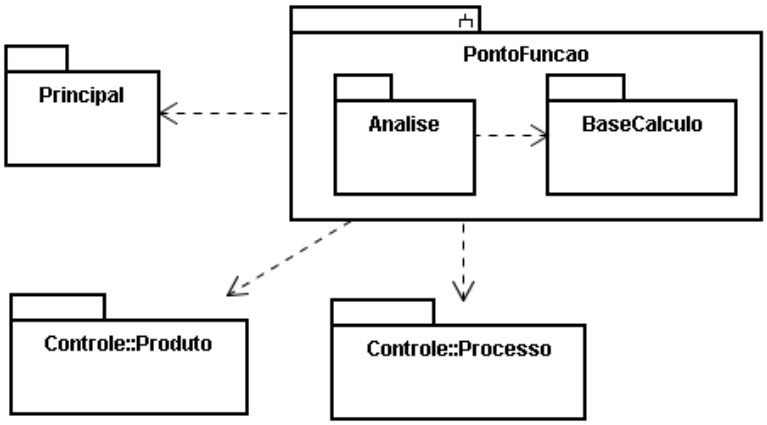

Figura A.7 - Diagrama de pacotes

Como já foi discutido, o pacote *PontoFuncao* possui as classes que materializam a ferramenta de Estimativa de Pontos de Função no ambiente. A seguir são apresentados os diagramas de classes da fase de projeto.

A.3.2 Pacote *Analise*

O pacote *Analise* contém as classes que materializam as funcionalidades definidas no documento de especificação de requisitos.

A.3.2.1 – Componente do Domínio do Problema (Cdp)

As classes do Cdp do pacote *Analise* são originadas a partir do modelo de análise, adequando as mesmas à tecnologia Java, usada na implementação. A Figura A.8 mostra o diagrama de classes desse pacote.

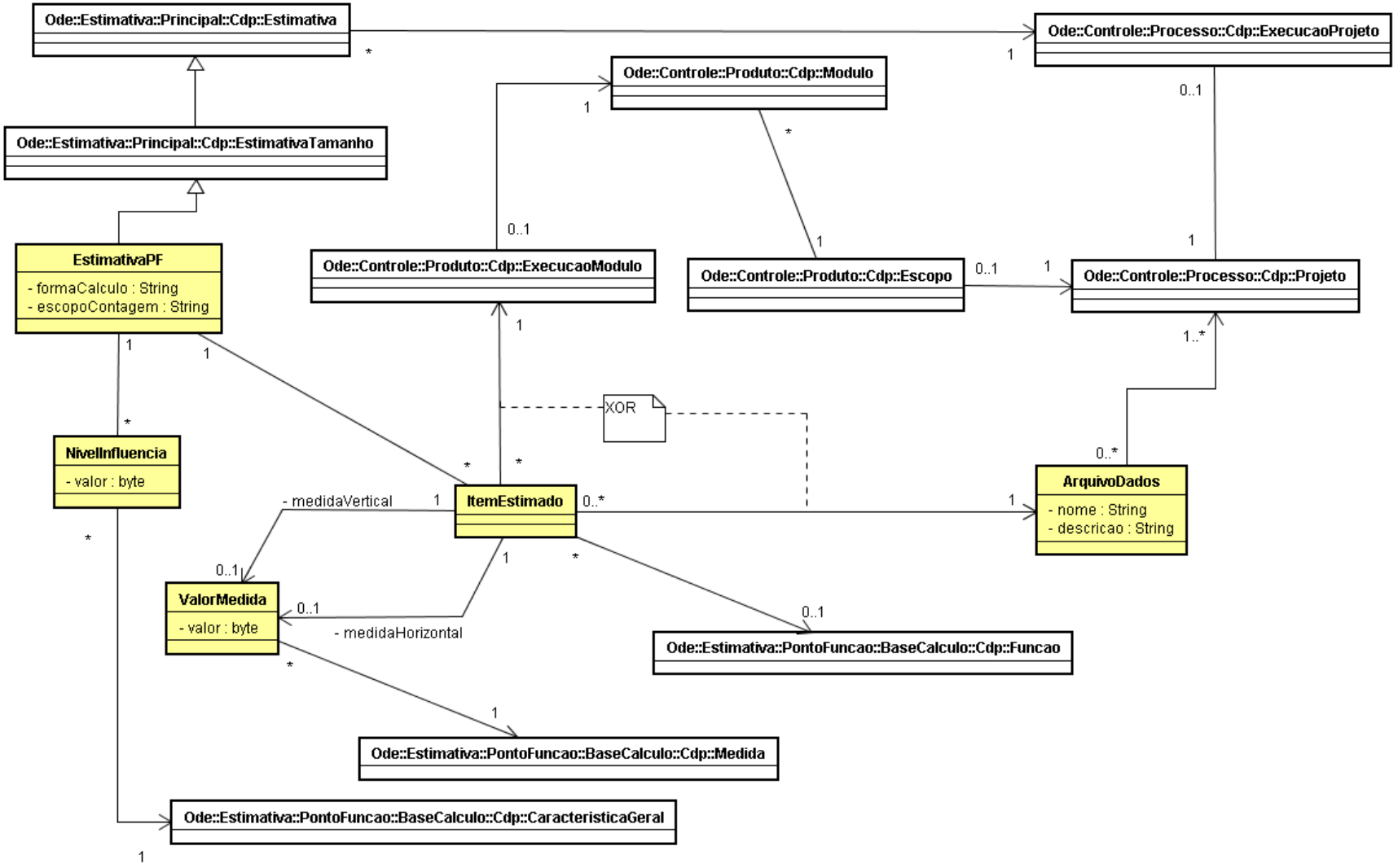

Figura A.8 – Componente de Domínio do Problema do pacote *Analise*

Apesar de não haver modificações significativas em relação ao modelo de análise quanto às classes do pacote (em amarelo), vale lembrar que, por Java não permitir herança múltipla, a hierarquia quanto ao escopo da estimativa foi eliminada e, portanto, a classe `EstimativaModulo` foi eliminada.

A.3.2.2 – Componente de Gerência de Dados (Cgd)

As classes contempladas no Cgd têm a função de abstrair o acesso ao banco de dados usado na implementação. A Figura A.9 mostra o diagrama de classes desse pacote

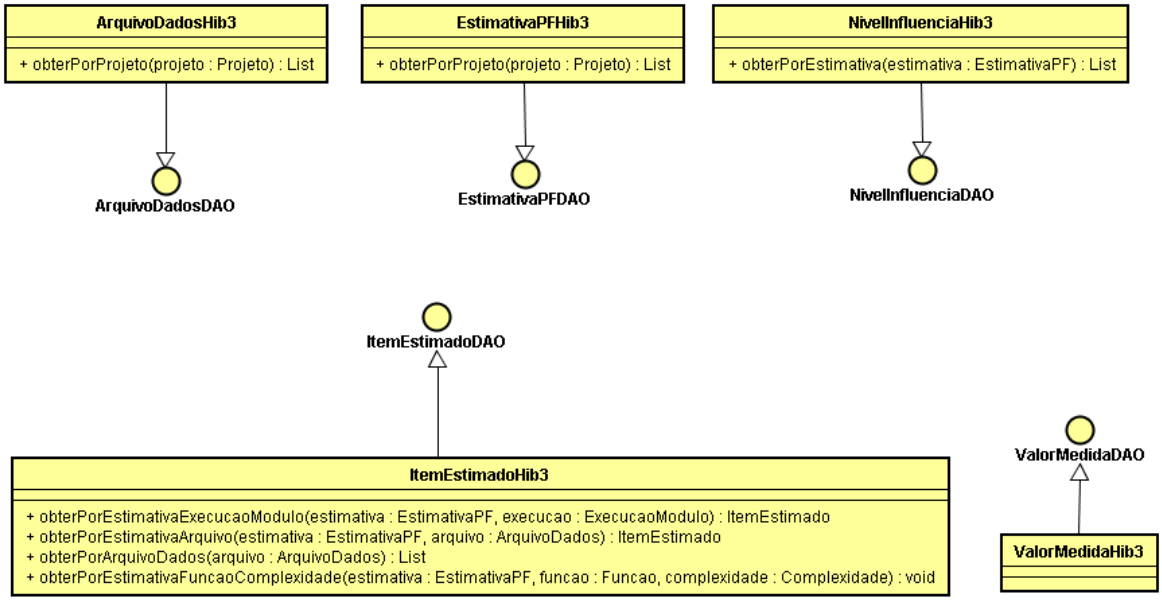

Figura A.9 - Componente de Gerência de Dados do pacote *Analise*

Todas as classes mostradas na Figura A.9 herdam da classe abstrata `Hibernate3DAO` que já implementa métodos básicos de manipulação de objetos, tais como *salvar*, *excluir*, *obterPorId* e *obterTodos*. Em alguns casos, foi necessário adicionar métodos auxiliares para recuperar objetos requeridos através de um objeto relacionado ao mesmo, tal como o método *obterPorArquivoDados* que retorna todos os objetos da classe `ItemEstimado` que se relacionam com um objeto `ArquivoDados`.

A.3.2.3 – Componente de Gerência de Tarefas (Cgt)

O componente de Gerência de Tarefas desse pacote possui uma única classe, que é responsável por realizar todos os casos de uso definidos no documento de requisitos, como mostra a Figura A.10.

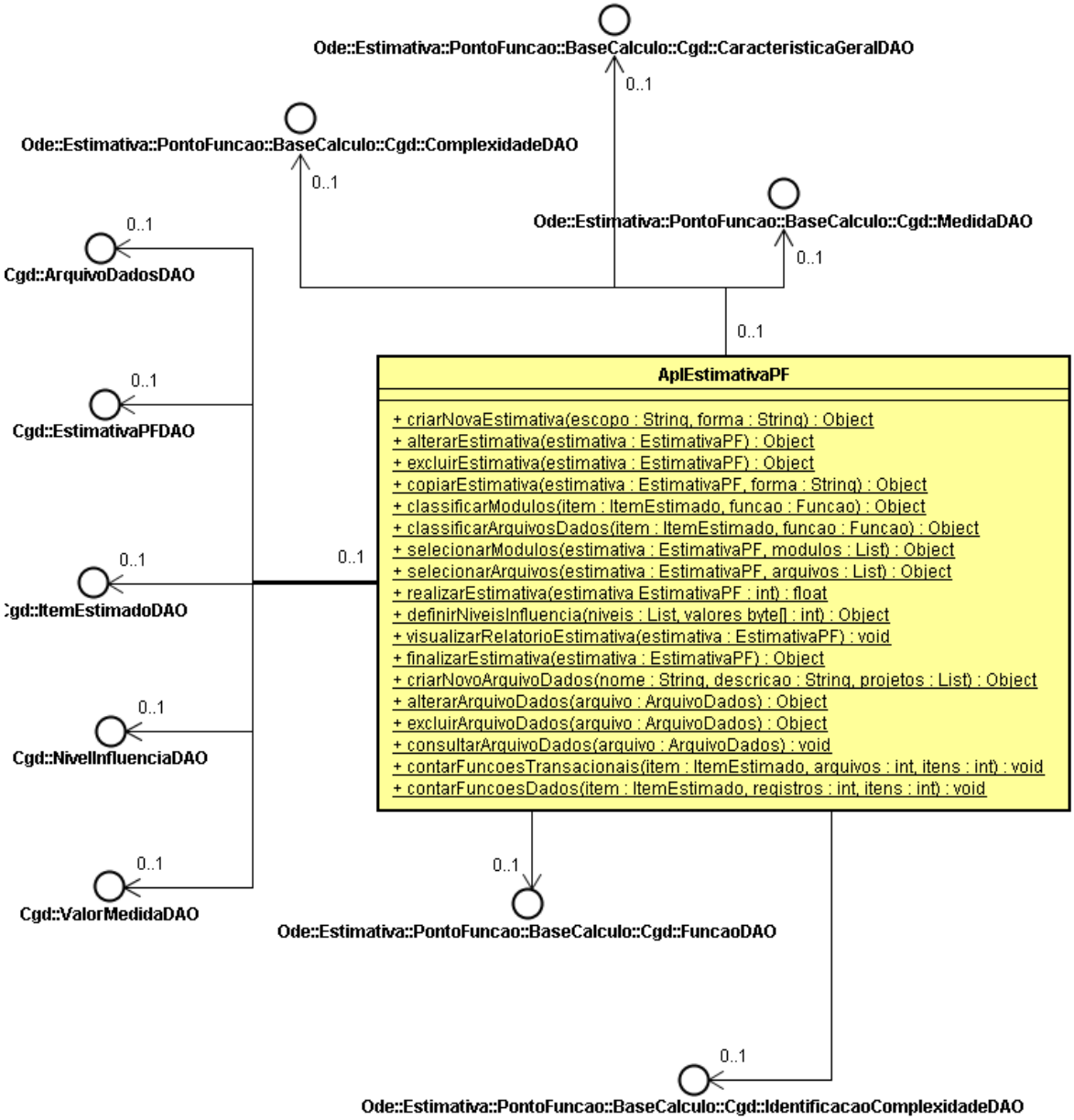

Figura A.10 - Componente de Gerência de Tarefas do Pacote *Analise*

O evento "Incluir Arquivo de dados no Projeto corrente" ainda não foi implementado nesta versão da ferramenta e, portanto, não tem um método associado.

A.3.2.4 – Componente de Interação Humana (Cih)

O componente de Interação Humana do pacote *Analise* materializa os cenários para realização dos casos de uso definidos no documento de especificação de requisitos. Para a implementação desse componente foram seguidos os novos padrões de interface para ferramentas internas do ambiente ODE. A Figura A.11 mostra a componente de interação humana relativa ao pacote *Analise.*

A janela de cadastro de estimativas (`JanCadastroEstimativaPF`) mostra as estimativas de pontos de função cadastradas no sistema e permite que se inicie a criação, alteração, exclusão, cópia e geração de relatório das mesmas. `JanRealizarEstimativaPF` agrupa e organiza a ordem de exibição dos painéis em tela, fazendo parte de um novo padrão para o ambiente ODE. O `PainelDefinirEscopo` permite que o gerente escolha entre os tipos de escopo de contagem e formas de cálculo contemplados na ferramenta, logo fazendo parte do evento *Criar Nova Estimativa*. `PainelCadastrarArquivo` atende ao caso de uso *Cadastrar Arquivo*, permitindo que sejam incluídos, alterados e excluídos arquivos no sistema.

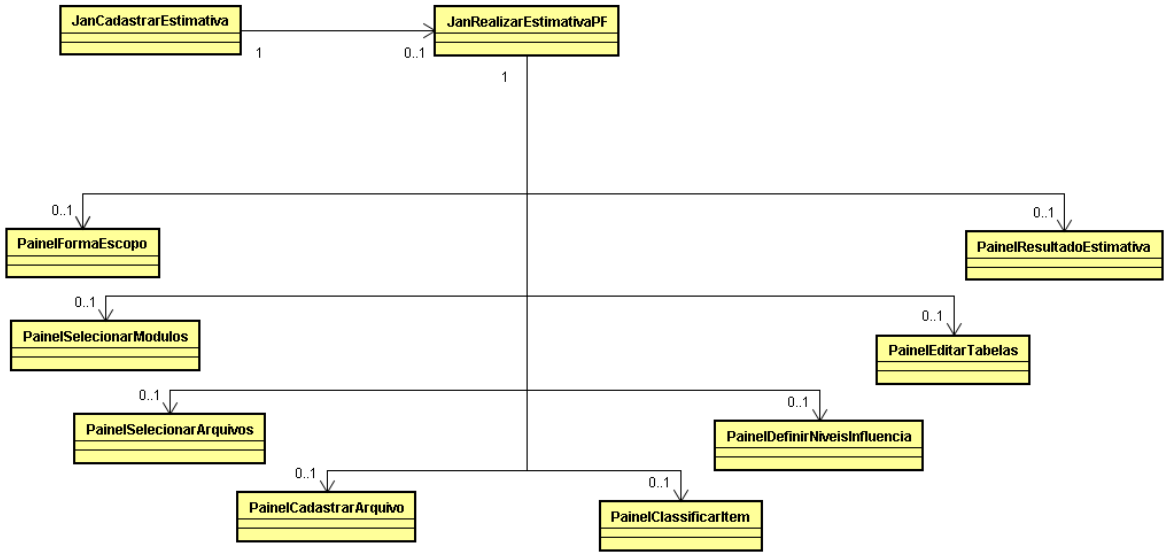

Figura A.11 - Componente de Interface Humana do pacote *Analise*

A seleção dos arquivos e módulos é contemplada pelas classes `PainelSelecaoArquivos` e `PainelSelecaoModulos`, respectivamente. `PainelClassificarItem` é responsável por permitir a classificação dos itens estimados. A definição dos níveis de

influência fica por conta do painel `PainelDefinirNiveisInfluencia`. A contagem das funções (caso de uso *Contar Funções*) é materializada pelo painel `PainelEditarTabelas`, que permite ao usuário definir o valor das medidas nos itens estimados e calcula a complexidade dos mesmos.

Finalmente o `PainelResultadoEstimativa` exibe um resumo da estimativa vigente, permitindo, ainda, que seja gerado o relatório referente ao evento *Visualizar Relatório de Estimativa.*

A.3.2.5 – Componente de Controle de Interação (Cci)

O Cci se responsabiliza por fazer o intercâmbio de dados entre a aplicação e a interface da ferramenta, como mostra a Figura A.12.

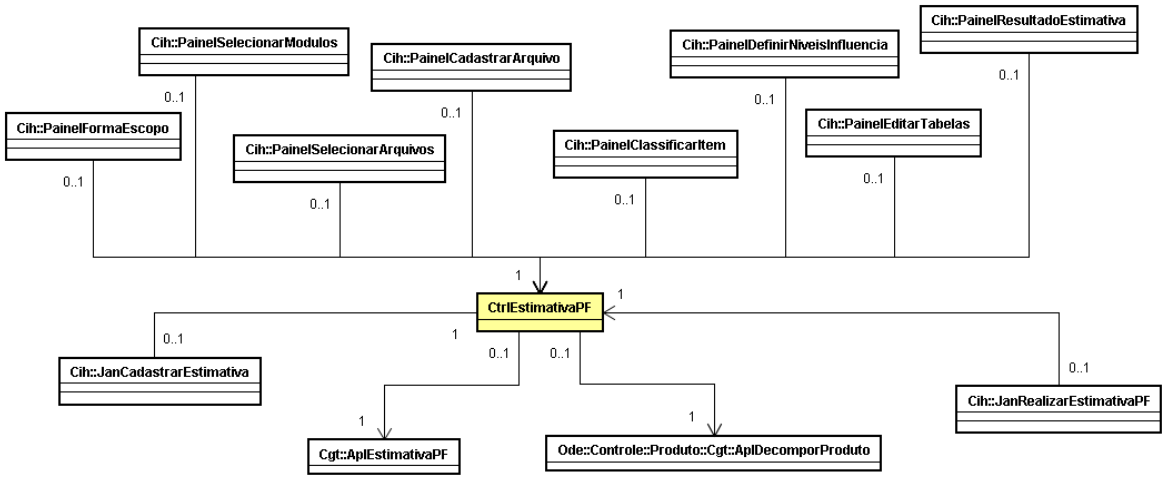

Figura A.12 - Componente de Controle de Interação do Pacote *Analise*

A navegabilidade no sentido `CtrlEstimativaPF` e `AplEstimativaPF` permite que as funcionalidades da ferramenta possam ser utilizadas por outras ferramentas. Além disso, se existir a necessidade de modificar a interface da ferramenta, basta reformular o controlador de acordo com mesma.

Em todos os momentos que um painel necessita de acessar alguma funcionalidade contemplada nas aplicações do modelo, ele o faz por meio do controlador `CtrlEstimativaPF`, que conhece as classes de aplicação capazes de atender à requisição da interface. Quando a aplicação requisitada dá um retorno, o controlador interpreta e atualiza as interfaces de acordo com a necessidade.

A.3.3 Pacote BaseCalculo

O pacote *BaseCalculo* contém as classes que apóiam a contagem dos pontos de função, contemplando todos os valores-padrão relativos ao método da Análise de Pontos de Função. O pacote *BaseCalculo* fica responsável apenas por guardar o domínio do problema e por gerenciar os dados relativos aos mesmos.

A.3.3.1 – Componente do Domínio do Problema (Cdp)

Não houve mudanças significativas entre o modelo de análise e o modelo relativo ao Cdp do pacote *BaseCalculo*, sendo adicionados apenas tipos de dados e navegabilidades das associações, como mostra a Figura A.13.

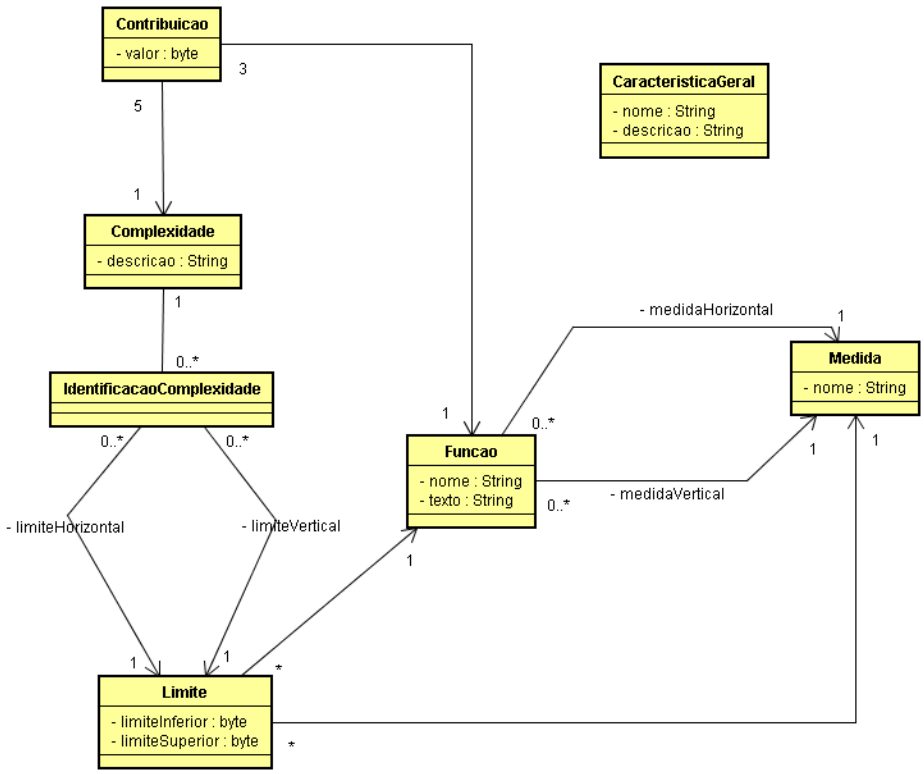

Figura A.13 - Componente de Domínio do Problema do Pacote *BaseCalculo*

A.3.3.2 – Componente de Gerência de Dados (Cgd)

Esse componente fica responsável por gerenciar os dados dos valores-padrão relativos ao método da Análise de Pontos de Função.

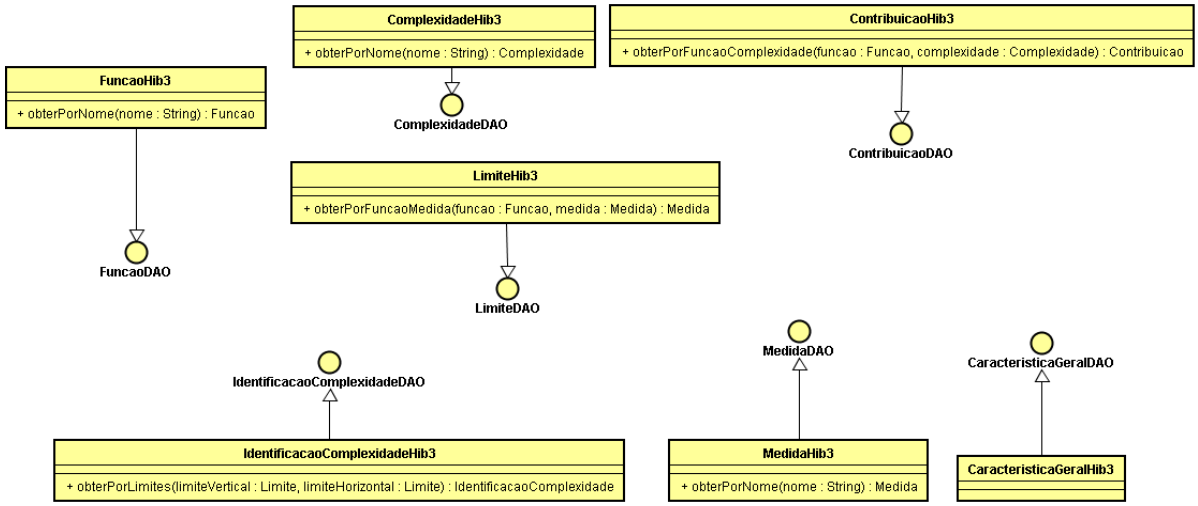

Figura A.14 - Componente de Gerência de Dados do Pacote *BaseCalculo*

A.3.4 A Ferramenta

São apresentadas nesta seção, algumas interfaces dos eventos mais relevantes da ferramenta, a saber Definição da Forma de Cálculo e Escopo de Contagem, e Contagem de Funções.

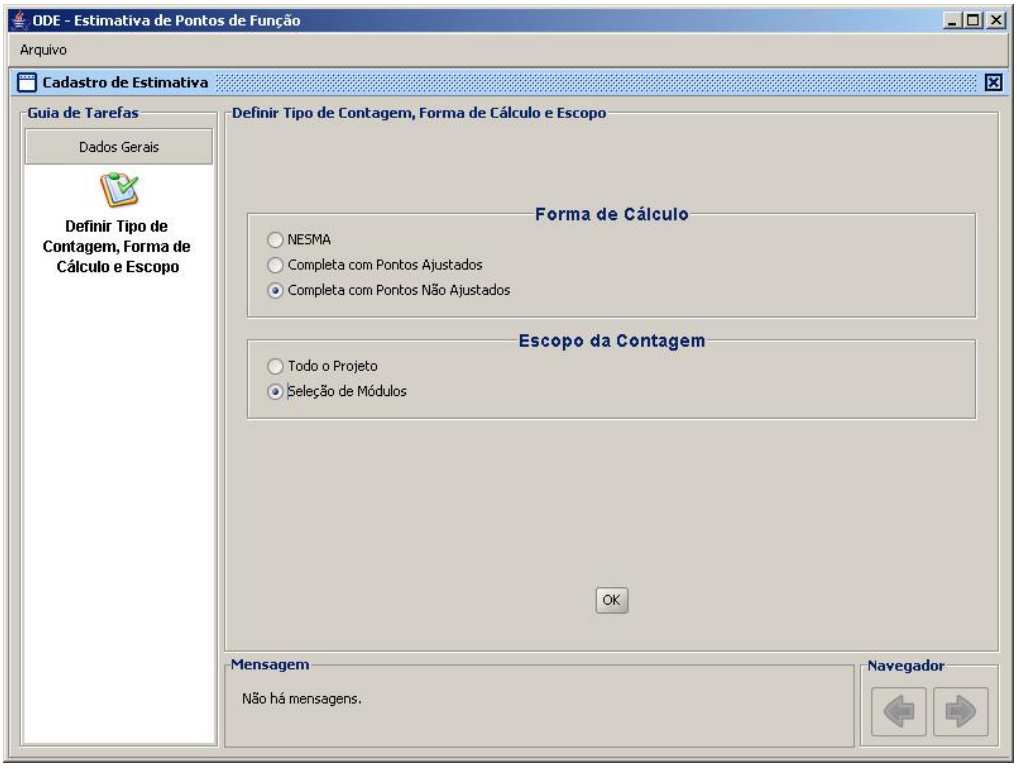

Figura A.15 - Criação de uma nova Estimativa de Pontos de Função

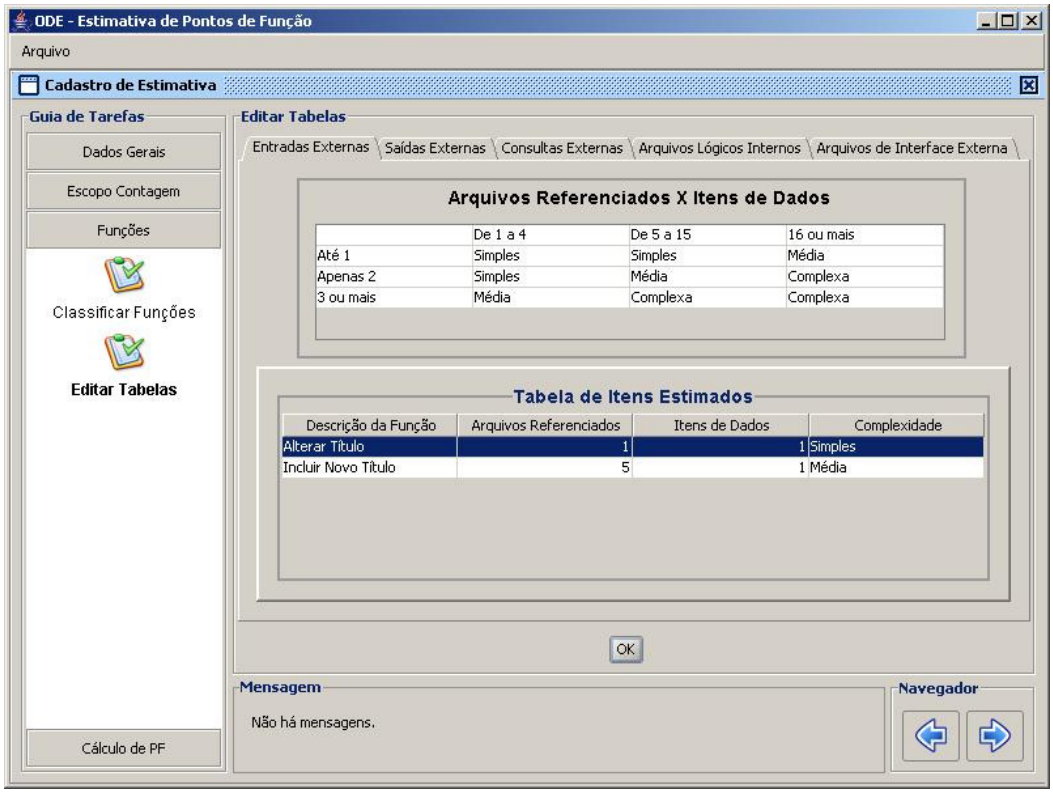

Figura A.16 – Contagem das funções.

Anexo B

Documentação da Evolução da Ferramenta de Apoio a Estimativas Usando Pontos de Caso de Uso

Este documento tem o intuito de prover informações sobre a versão atual da ferramenta de Estimativas de Pontos de Caso de Uso, abrangendo todos as fases do ciclo de vida de desenvolvimento.

Inicialmente é apresentada a especificação de requisitos da ferramenta (seção B.1), detalhando apenas os requisitos funcionais, já que os requisitos não-funcionais foram tratados na seção 4.3. Em seguida, na seção B.2, é discutida a fase de análise da ferramenta apresentando uma visão do domínio do problema sem levar em conta qualquer aspecto tecnológico. Finalmente a seção B.3 trata do projeto dessa versão da ferramenta.

B.1 Especificação de Requisitos

Nesta seção é apresentada a Especificação de Requisitos da ferramenta de Estimativa de Pontos de Caso de Uso, iniciando com uma descrição do problema (seção B.1.1) e em seguida apresentando a modelagem dos casos de uso levados em conta e as descrições dos mesmos (seção B.1.2).

B.1.1 Descrição do Mini-Mundo

A Análise dos Pontos de Caso de Uso (APCU) é um método de estimativa de tamanho de projetos de software orientados a objetos, baseada nas técnicas de Análise de Pontos de Função e Modelagem de Casos de Uso (ANDRADE, 2004). O método explora o modelo e a descrição dos casos de uso, retirando informações necessárias para a confecção da estimativa. Basicamente, a APCU envolve os seguintes passos: (i) contagem e classificação dos atores e casos de uso segundo tabelas pré-definidas; (ii) cálculo dos Pontos de Caso de Uso (PCUs) não ajustados; (iii) determinação dos fatores de complexidade técnica e ambiental; (iv) determinação dos PCUs ajustados.

A primeira versão da ferramenta de Estimativas de Ponto de Caso de Uso em ODE foi desenvolvida em (LAHAS, 2005). Até então a ferramenta permitia a contagem de PCUs para projetos de desenvolvimento sob um foco do projeto por completo, não sendo possível elaborar contagens para uma certa seleção de casos de uso e atores. Os dados consumidos e gerados pela mesma não eram integrados com a ferramenta de modelagem UML do ambiente ODE – a ferramenta OODE (CARREIRA, 2003), que permite a criação de modelos de objetos (diagramas de caso de uso, classes etc), além da manipulação dos elementos de modelo. Além disso, a ferramenta contemplava casos de uso referentes à edição de fatores de peso de casos de uso, prática desaconselhada por causar confusão em um momento de comparação de pontos de casos de uso.

A Figura B.1 mostra o diagrama de pacotes da ferramenta. No pacote *Principal* estão os elementos gerais utilizados na realização de estimativas em ODE, tanto sob o contexto de escopo (projeto, módulo etc) quanto de natureza da estimativa (tamanho, esforço, custo e tempo). Já o subsistema *PontoCasoUso* diz respeito especificamente ao apoio à Estimativa de Pontos de Caso de Uso (PCUs). Esse subsistema possui dois pacotes:

o pacote *EstimativaPCU*, que trata das funcionalidades ligadas diretamente à contagem de PCUs, e o pacote *BaseCalculo* anteriormente chamado de *FatoresAjuste*, que se refere às funcionalidades de apoio à definição de fatores de peso dos atores e casos de uso, além de contemplar as características padronizadas pela metodologia.

No pacote *Ode::UML::ElementosComportamentais::CasoUso* encontram-se os elementos referentes à modelagem de casos de uso. A dependência entre o último e *Ode::UML::Fundação::Nucleo* e *Ode::Artefato::Modelo* é relativa à ferramenta OODE, que visa permitir a criação e manipulação de modelos de objetos e seus elementos.

Neste documento é discutido apenas o pacote *EstimativaPCU*, lembrando-se de que os casos de uso contemplados no pacote *BaseCalculo,* referentes à edição de peso de ator e caso de uso, foram removidos por não apresentar valor significativo em contagens de estimativa do presente momento.

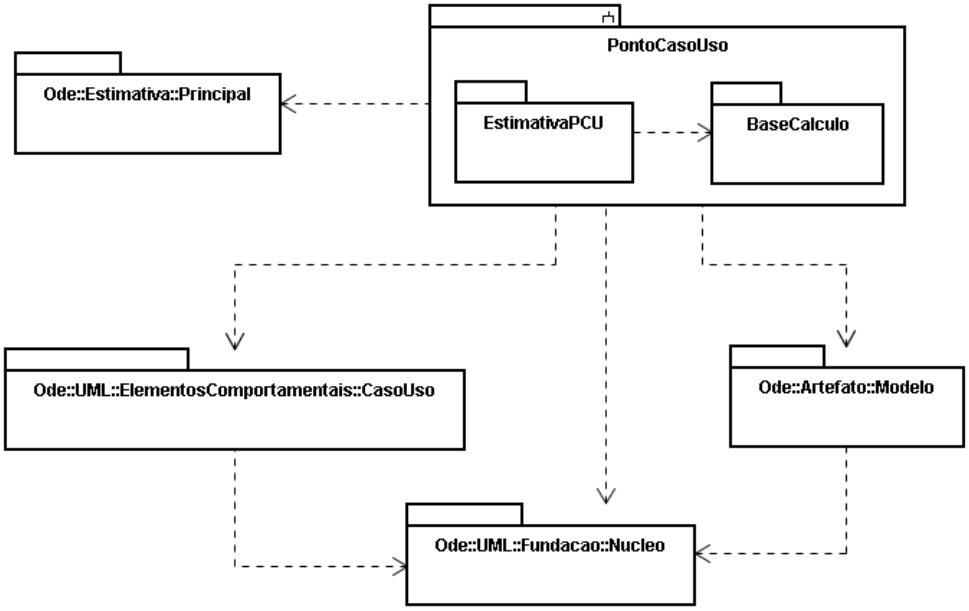

Figura B.1 - Diagrama de Pacotes

B.1.2 Modelo de Casos de Uso

A Figura B.1 mostra o diagrama de casos de uso do pacote *EstimativaPCU*. Os casos de uso mostrados nessa figura são provenientes da re-estruturação da especificação proposta em (LAHAS, 2005).

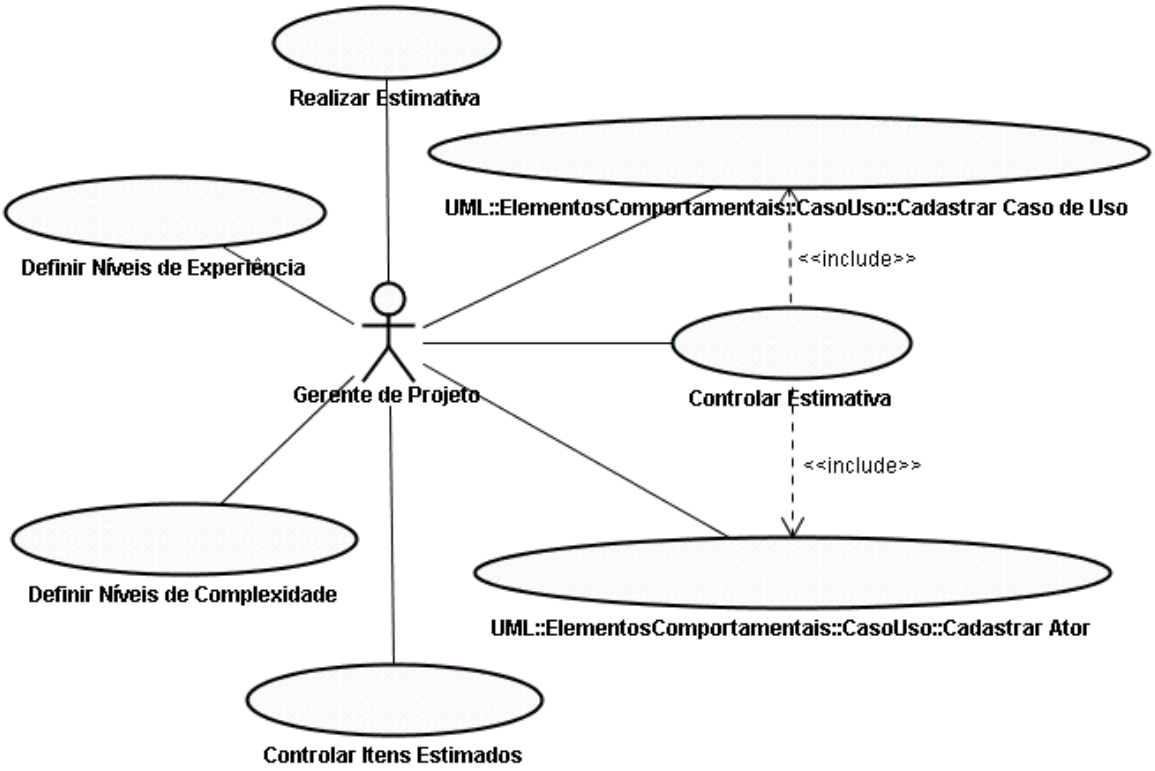

Figura B.13 - Diagrama de Casos de Uso do pacote EstimativaPCU

Ao se iniciar uma estimativa, é necessário que se defina o escopo da contagem, definindo se a mesma é para o projeto como um todo ou se é para uma seleção de casos de uso. Ao se escolher o escopo de seleção de casos de uso, é interessante permitir que o gerente de projeto possa incluir, alterar ou excluir casos de uso e atores.

A seguir, os casos de uso propostos são descritos.

Sub-Sistema: *PontoCasoUso* (pacote *EstimativaPCU*)

Caso de Uso: Controlar Estimativa

Descrição: Este caso de uso permite criar uma nova estimativa de Pontos de Caso de Uso para o projeto corrente, abrir, copiar ou excluir uma estimativa existente do projeto corrente.

Curso Normal:

Criar Nova Estimativa

O gerente informa o escopo da contagem da nova estimativa: seleção de casos de uso ou todo o projeto. Os dados são registrados juntamente com a data de criação. Caso o escopo seja de alguns casos de uso, permite-se cadastrar atores e casos de uso, realizando-se os casos de uso *Cadastrar Ator* e *Cadastrar Caso de Uso*.

Alterar Estimativa

O gerente informa a estimativa ainda não finalizada que deseja abrir. Os dados da estimativa são exibidos e o gerente pode editar a mesma.

Copiar Estimativa:

O gerente informa a estimativa finalizada que deseja copiar. Um nova estimativa é criada com os dados da estimativa selecionada.

Excluir Estimativa:

O gerente informa a estimativa que deseja excluir. Uma confirmação é requisitada e, caso confirmada, a estimativa é excluída.

Visualizar Relatório de Estimativa

O gerente informa a estimativa que deseja visualizar um relatório. São exibidos os dados daquela estimativa em forma de relatório.

Finalizar Estimativa

O gerente informa a estimativa que deseja finalizar. Uma mensagem de confirmação é exibida, mostrando que uma estimativa finalizada não poderá mais ser alterada. Após a confirmação, a data de finalização da estimativa é registrada.

Curso Alternativo:

Alterar Estimativa:

A estimativa selecionada está finalizada: uma mensagem é exibida informando que não é possível alterar estimativas finalizadas.

Classes: EstimativaPCU, Projeto, FatorPesoAtor, FatorPesoCasoUso, ClassificacaoAtor, ClassificacaoCasoUso, NivelComplexidade, NivelExperiencia, CaracteristicaExperiencia, CaracteristicaComplexidade.

Sub-Sistema: *PontoCasoUso* (pacote *EstimativaPCU*)

Caso de Uso: Controlar Itens Estimados

Descrição: Este caso de uso permite selecionar casos de uso e classificar os itens a serem contados em uma estimativa (atores e casos de uso).

Curso Normal:

Selecionar Casos de Uso:

Pré-condição: O escopo da contagem não pode ser o projeto como um todo.

São exibidos todos os casos de uso registrados no projeto. O gerente de projeto seleciona os casos de uso que deseja incluir na contagem. Os atores associados aos casos de uso selecionados são também incluídos na contagem. Os dados são registrados.

Classificar Ator:

São apresentados os atores que fazem parte do escopo da contagem. O gerente classifica cada ator segundo seu tipo, a saber: Simples, Médio e Complexo. A classificação é registrada para cada ator.

Classificar Casos de Uso:

São apresentados os casos de uso que fazem parte do escopo da contagem. O gerente classifica cada caso de uso segundo seu tipo, a saber: Simples, Médio e Complexo. A classificação é registrada para cada caso de uso.

Classes: EstimativaPCU, Ator, CasoUso, ClassificacaoCasoUso, ClassificacaoAtor, FatorPesoCasoUso, FatorPesoAtor.

Sub-Sistema: *CasoUso* (pacote *UML::ElementosComportamentais*[3])

Caso de Uso: Cadastrar Ator

Descrição: Este caso de uso permite criar, excluir, alterar e consultar atores de um projeto.

Curso Normal:

Criar Novo Ator:

O gerente informa o nome e descrição do novo ator. Os dados são validados e um novo ator é criado associado ao modelo de objetos do projeto corrente.

Alterar Ator:

O gerente informa o ator que deseja alterar e os novos dados. Os dados são validados e registrados.

Consultar Ator:

O gerente informa o ator que deseja consultar. Os dados são apresentados.

Excluir Ator:

O gerente informa o ator que deseja excluir. Uma solicitação de confirmação é exibida e, caso confirmada, o ator é excluído. Só é permitida a exclusão de atores que não estejam relacionados com nenhuma contagem.

Curso Alternativo:

Excluir Ator:

O ator está sendo utilizado em alguma contagem: Uma mensagem é exibida, informando que o ator em questão não poderá ser excluído, pois existem uma ou mais contagens relacionadas a ele.

Classes: Ator, ModeloObjetos, ClassificacaoAtor

[3] O pacote *UML::ElementosComportamentais* se refere à modelagem UML e foi definido no contexto da ferramenta de apoio à UML de ODE, denominada OODE (CARREIRA, 2003).

Sub-Sistema: *CasoUso* (pacote *UML::ElementosComportamentais*)
Caso de Uso: Cadastrar Caso de Uso
Descrição: Este caso de uso permite criar, excluir, alterar e consultar casos de uso de um projeto.

Curso Normal:

Criar Novo Caso de Uso:

O gerente informa o nome e descrição do novo caso de uso, eventos e atores associados. Os dados são validados e um novo caso de uso é criado associado ao modelo de objetos do projeto corrente, juntamente com seus eventos.

Alterar Caso de Uso:

O gerente informa o caso de uso que deseja alterar e os novos dados. Os dados são validados e registrados.

Consultar Caso de Uso:

O gerente informa o caso de uso que deseja consultar. Os dados são apresentados.

Excluir Caso de Uso:

O gerente informa o caso de uso que deseja excluir. Uma solicitação de confirmação é exibida e, caso confirmada, o caso de uso é excluído. Só é permitida a exclusão de casos de uso que não estejam relacionados com nenhuma contagem.

Curso Alternativo:

Excluir Caso de Uso:

O caso de uso está sendo utilizado em alguma contagem: Uma mensagem é exibida, informando que o caso de uso em questão não poderá ser excluído, pois existem uma ou mais contagens relacionadas a ele.

Classes: CasoUso, EventoCasoUso, ModeloObjetos, ClassificacaoCasoUso.

Sub-Sistema: *PontoCasoUso* (pacote *EstimativaPCU*)

Caso de Uso: Definir Níveis de Complexidade

Descrição: Este caso de uso é responsável pela definição dos níveis de complexidade de um projeto, tomando por base as treze características apontadas pela técnica de PCU (vide Capítulo 2).

Curso Normal:

O Gerente de Projeto seleciona o projeto e informa, para cada uma das treze características abaixo relacionadas, o seu nível de complexidade, que deve ser um valor entre 0 - nenhuma influência e 5 - forte influência:

1. O quanto o sistema é distribuído
2. Performance de entrada do programa
3. Usuário eficiente
4. Processamento de dados complexo
5. Código deve ser reusado
6. Fácil de instalar o programa
7. Fácil de usar o programa
8. Portabilidade
9. Fácil de mudar o código, legível
10. Muitas pessoas podem usar ao mesmo tempo o programa
11. Inclui código de segurança especial
12. Provê acesso direto com os usuários
13. Treinamento especial é exigido para orientar o usuário.

Caso os dados sejam válidos, os graus de complexidade são registrados.

Curso Alternativo:

Caso o Gerente de Projeto não informe um dos 13 graus de complexidade, uma mensagem solicitando a entrada do dado é mostrada. Se um dos graus de complexidade informados não estiver na faixa de valor de 0 a 5, então uma mensagem de erro é exibida, solicitando a correção da informação.

Classes: EstimativaPCU, NivelComplexidade, CaracteristicaComplexidade.

Sub-Sistema: *PontoCasoUso* (pacote *EstimativaPCU*)

Caso de Uso: Definir Níveis de Experiência

Descrição: Este caso de uso é responsável pela definição dos níveis de experiência de um projeto, tomando por base as sete características definidas pela técnica de PCU (vide Capítulo 2).

Curso Normal:

O Gerente de Projeto seleciona o projeto e informa, para cada uma das sete características abaixo relacionadas, o nível de experiência no projeto, que deve ser um valor entre 0 - nenhuma influência e 5 - forte influência:

1. Equipe com programadores que não se conhecem
2. A maioria da equipe não sabe programar na linguagem
3. Bons programadores compõem a equipe
4. Equipe com programadores rápidos
5. Sem expectativa de mudança no código
6. Equipe sem pessoas que dedicam pouco tempo
7. Dificuldade de programação da linguagem

Caso os dados sejam válidos, os graus de experiência são registrados.

Curso Alternativo:

Caso o Gerente de Projeto não informe um dos sete graus de experiência, uma mensagem solicitando a entrada do dado é mostrada. Se um dos graus de experiência informados não estiver na faixa de valor de 0 a 5, então uma mensagem de erro é exibida, solicitando a correção de informação.

Classes: EstimativaPCU, NivelExperiencia, CaracteristicaExperiencia.

Sub-Sistema: *PontoCasoUso* (pacote *EstimativaPCU*)

Caso de Uso: Realizar Estimativa

Descrição: Este caso de uso calcula os pontos de caso de uso da estimativa.

Curso Normal:

Pré-condição: Níveis de complexidade e experiência definidos, atores e casos de uso classificados.

O Gerente de Projeto seleciona o projeto para o qual deseja calcular os pontos de caso de uso. A seguir, os seguintes passos são realizados:

1. *Cálculo dos Pontos de Caso de Uso Não Ajustados*

Para calcular os pontos de caso de uso não ajustados (PCUNA), é necessário calcular o peso dos atores (PA) e o peso dos casos de uso (PT) envolvidos no projeto, seguindo as seguintes fórmulas:

$$PA = \sum_{i=1}^{3} vlAtor_i * qteAtor_i$$

$$PT = \sum_{i=1}^{3} vlTransacao_i * qteTransacao_i$$

Uma vez calculados os pesos dos atores e dos casos de uso, calculam-se os pontos de caso de uso não ajustados (PCUNA), segundo a seguinte fórmula:

$$PCUNA = PA + PT$$

2. *Cálculo do Fator de Ajuste de Complexidade Técnica*

Com base nos graus de influência de complexidade (GICs) das treze características definidas pela técnica PCU, multiplicados pelos níveis de influência de complexidade (NIC), calcula-se o nível de influência de complexidade total (NICT), segundo a seguinte fórmula:

$$NICT = \sum_{i=1}^{13} (GIC_i * NIC_i)$$

O valor do fator de ajuste de complexidade técnica (FTC) é determinado, então, pela fórmula:

$$FTC = (NICT * 0,01) + 0,65$$

3. Cálculo do Fator de Ajuste Ambiental

Com bases nos graus de influência ambientais (GIAs) das sete características definidas pela técnica PCU, multiplicados pelos níveis de influência ambiental (NIA), calcula-se o nível de influência ambiental total (NIAT), segundo a seguinte fórmula:

$$NIAT = \sum_{i=1}^{7} (GIA_i * NIA_i)$$

O valor do fator de ajuste ambiental (FA) é determinado, então, pela fórmula:

$$FA = (NIAT * 0,01) + 0,65$$

4. Cálculo dos Pontos de Caso de Uso

O total de pontos de caso de uso (PCU) é calculado pela fórmula:

$$PCU = PCUNA * FTC * FA$$

Finalmente, são mostrados para o Gerente de Projeto os resultados de todas as etapas do cálculo.

Classes: EstimativaPCU, ClassificacaoAtor, ClassificacaoCasoUso, NivelExperiencia, NivelComplexidade, FatorPesoAtor, FatorPesoCasoUso, CaracteristicaComplexidade, CaracteristicaExperiencia.

B.2 Análise

Nesta seção é apresentada a Especificação de Análise do projeto de manutenção da ferramenta de Estimativas de Pontos de Caso de Uso.

Na seção B.2.1, é apresentado um diagrama de pacotes mostrando o relacionamento entre os pacotes do ambiente ODE e os pacotes específicos da ferramenta-alvo. Em seguida, a seção B.2.2 apresenta e discute os diagramas de classes derivados a partir da especificação de requisitos.

B.2.1 Modelo de Classes

O diagrama de pacotes da Figura B.3 mostra as dependências existentes entre os pacotes contemplados nessa ferramenta.

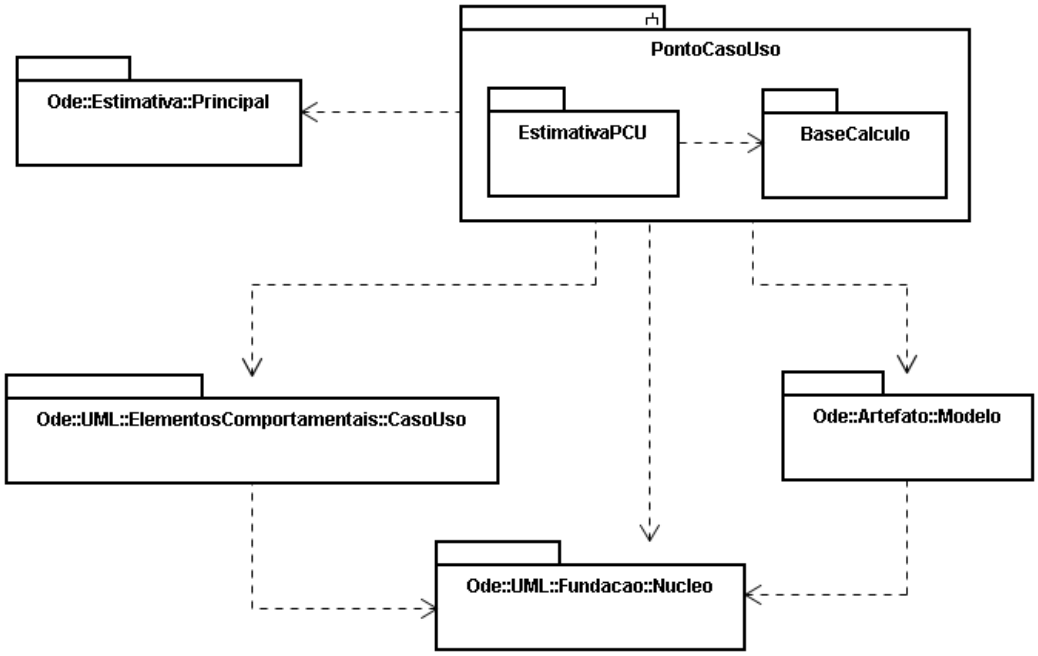

Figura B.3 - Diagrama de pacotes

O pacote *Principal* contém a infra-estrutura utilizada nas ferramentas de estimativa de ODE. O subsistema *PontoCasoUso* diz respeito especificamente ao apoio à Estimativa de Pontos de Caso de Uso (PCUs). Esse subsistema possui dois pacotes: o pacote *EstimativaPCU*, que trata das funcionalidades ligadas diretamente à contagem de PCUs, e o pacote *BaseCalculo*, que comporta as classes que guardam informações

padronizadas estabelecidas pela metodologia. A seguir são apresentados os diagramas de classe da fase de análise para os pacotes *EstimativaPCU* e *BaseCalculo*.

B.2.2 Diagramas de Classes

A seguir são apresentados os diagramas de classe dos pacotes *EstimativaPCU* e *BaseCalculo*. É válido lembrar que o trabalho de reestruturação foi feito com o foco sobre o pacote *EstimativaPCU*.

B.2.2.1 – Pacote EstimativaPCU

O modelo da Figura B.4 mostra uma infra-estrutura em que diferentes tipos de estimativas, organizados pela natureza da estimativa sendo realizada (atualmente, de esforço e tamanho), ou pelo escopo da estimativa (atualmente, para o projeto como um todo, para um módulo do projeto ou para atividades específicas), são abrigados como especializações da classe `Estimativa`, conforme definido no pacote *Principal*.

O escopo de contagem de uma estimativa de pontos de caso de uso (o projeto como um todo ou apenas alguns casos de uso) é registrado na classe `EstimativaPCU`, que representa uma contagem de estimativa de pontos de caso de uso. Cada objeto dessa classe está relacionado diretamente com as classificações de ator e caso de uso da contagem. Cada classificação está associada a um fator de peso que representa o valor da complexidade do respectivo caso de uso ou ator definido pelo gerente de projeto com base nas tabelas propostas pelo método.

Quando se deseja obter PCUs ajustados, é necessário definir os valores de fatores técnicos a serem usados para ajustar os PCUs. Assim, uma estimativa pode ter sete níveis de experiência definidos pelo gerente, sendo que cada nível de experiência está diretamente relacionado a uma característica de experiência, que registra o grau de influência da mesma, conforme estabelecido pelo método. De maneira análoga, uma estimativa de PCUs pode ter treze níveis de complexidade, sendo que cada nível de complexidade registra o grau de influência de cada uma das treze características de complexidade definidas pelo método.

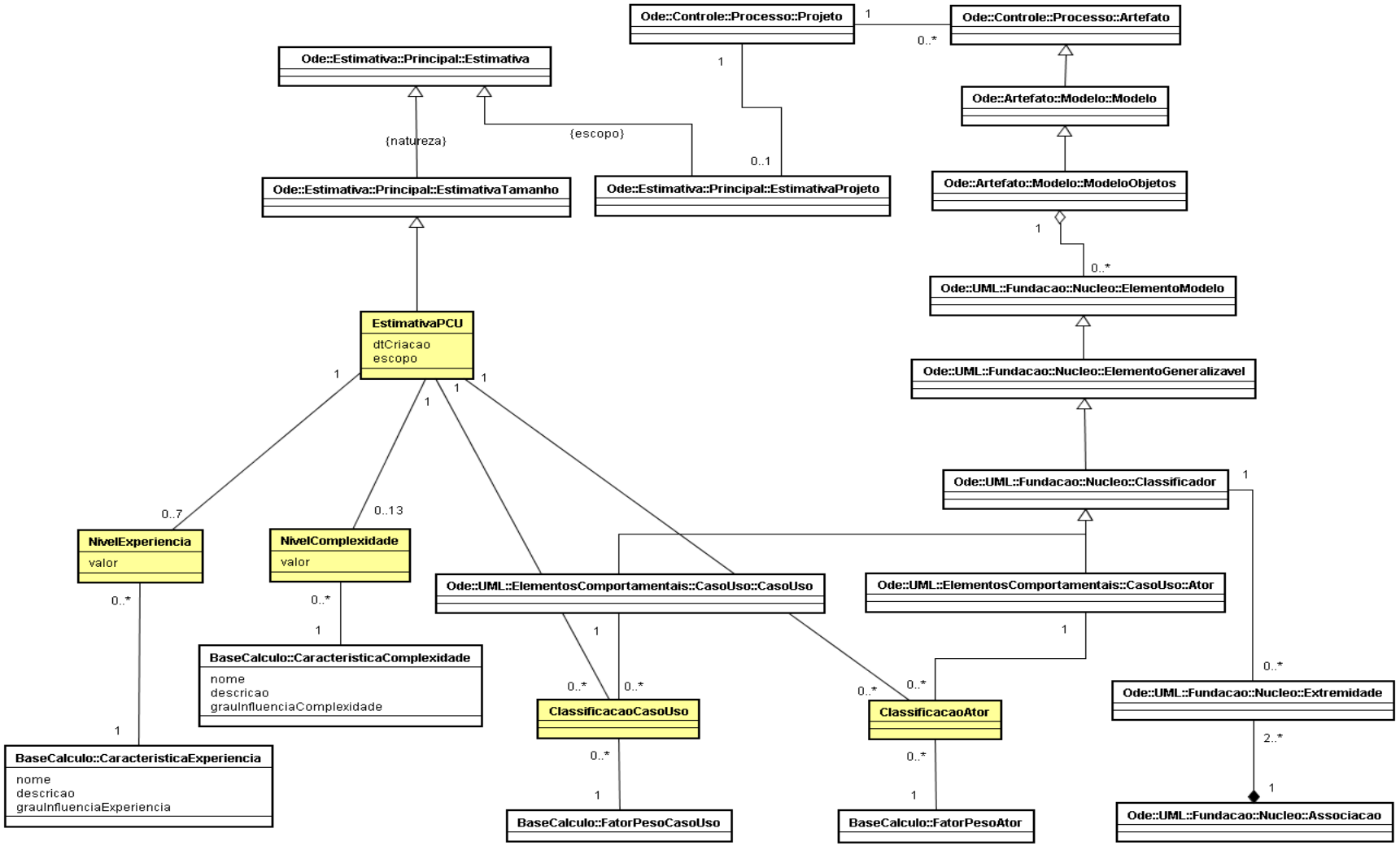

Figura B.4 - Diagrama de Classes do pacote EstimativaPCU

B.2.2.2 – Pacote BaseCalculo

A técnica de PCUs utiliza fatores de ajuste de pesos dos casos de uso e atores. Tanto casos de uso quanto atores podem ser definidos como simples, médios ou complexos. Para cada tipo, um valor de fator de peso é definido. Além disso, para os fatores de ajuste de peso de casos de uso, deve-se informar o número de transações relacionadas a um fator (LAHAS, 2005). As características referentes aos fatores técnicos definidas pelo método são registradas pelas classes `CaracteristicaComplexidade` e `CaracteristicaExperiencia`. A Figura B.5 apresenta o diagrama de classes desse pacote.

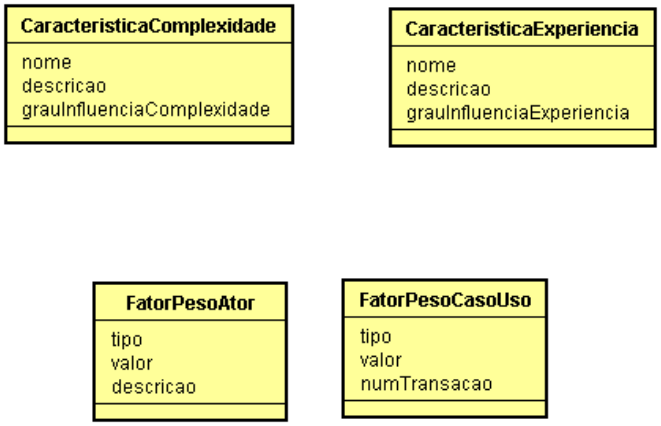

Figura B.5 - Diagrama de Classes do pacote BaseCalculo

B.3 Projeto e Implementação

Nesta seção é apresentada a Especificação de Projeto do projeto de manutenção da ferramenta de Estimativas de Pontos de Caso de Uso.

Nesse ponto do projeto todos os resultados obtidos na fase de análise são expostos aos aspectos tecnológicos adotados no ambiente ODE, tais como a linguagem de programação, mecanismo de persistência adotado e arquitetura de componentes adotada.

A seção B.3.1 apresenta a organização física dos pacotes que compõem a ferramenta e o relacionamento entre as classes da mesma e as classes do ambiente. Em seguida as seções B.3.2 e B.3.3 apresentam e discutem os diagramas de classes dos pacotes encontrados na ferramenta para cada componente definido na arquitetura. Finalmente a seção B.3.4 apresenta algumas interfaces da ferramenta.

B.3.1 Organização dos Pacotes

O diagrama de pacotes da Figura B.6 mostra as dependências existentes entre os pacotes físicos contemplados nessa ferramenta.

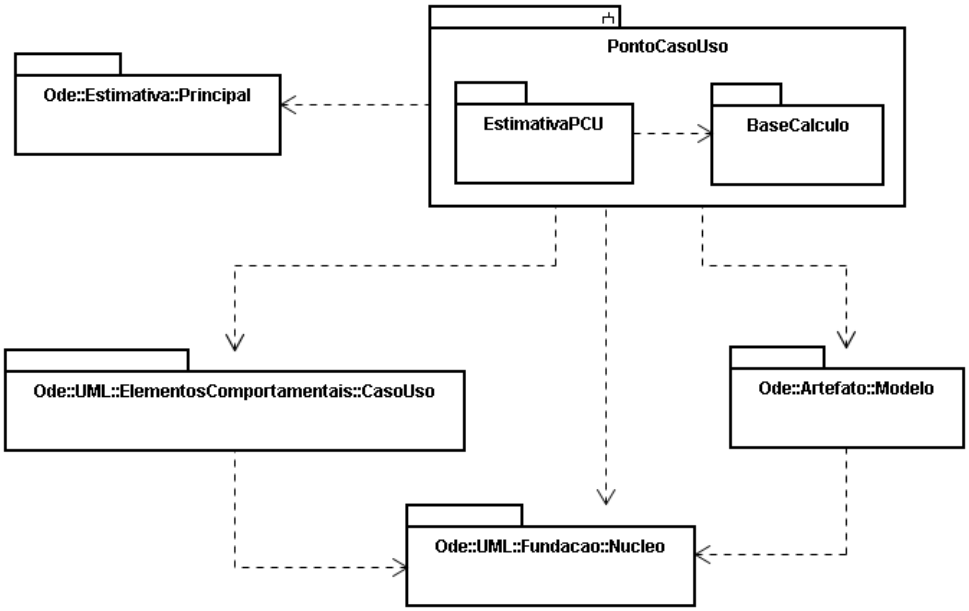

Figura B.6 - Diagrama de pacotes

Como já foi discutido, o pacote *EstimativaPCU* possui as classes que materializam a ferramenta de Estimativa de Pontos de Caso de Uso no ambiente. A seguir são apresentados os diagramas de classes da fase de projeto.

B.3.2 Pacote EstimativaPCU

O pacote *EstimativaPCU* contém as classes que tratam as funcionalidades definidas no documento de especificação de requisitos.

B.3.2.1 – Componente do Domínio do Problema (Cdp)

As classes do Cdp do pacote *EstimativaPCU*, mostrado na Figura B.7, foram originadas a partir do modelo de análise, adequando as mesmas à tecnologia Java, usada na implementação.

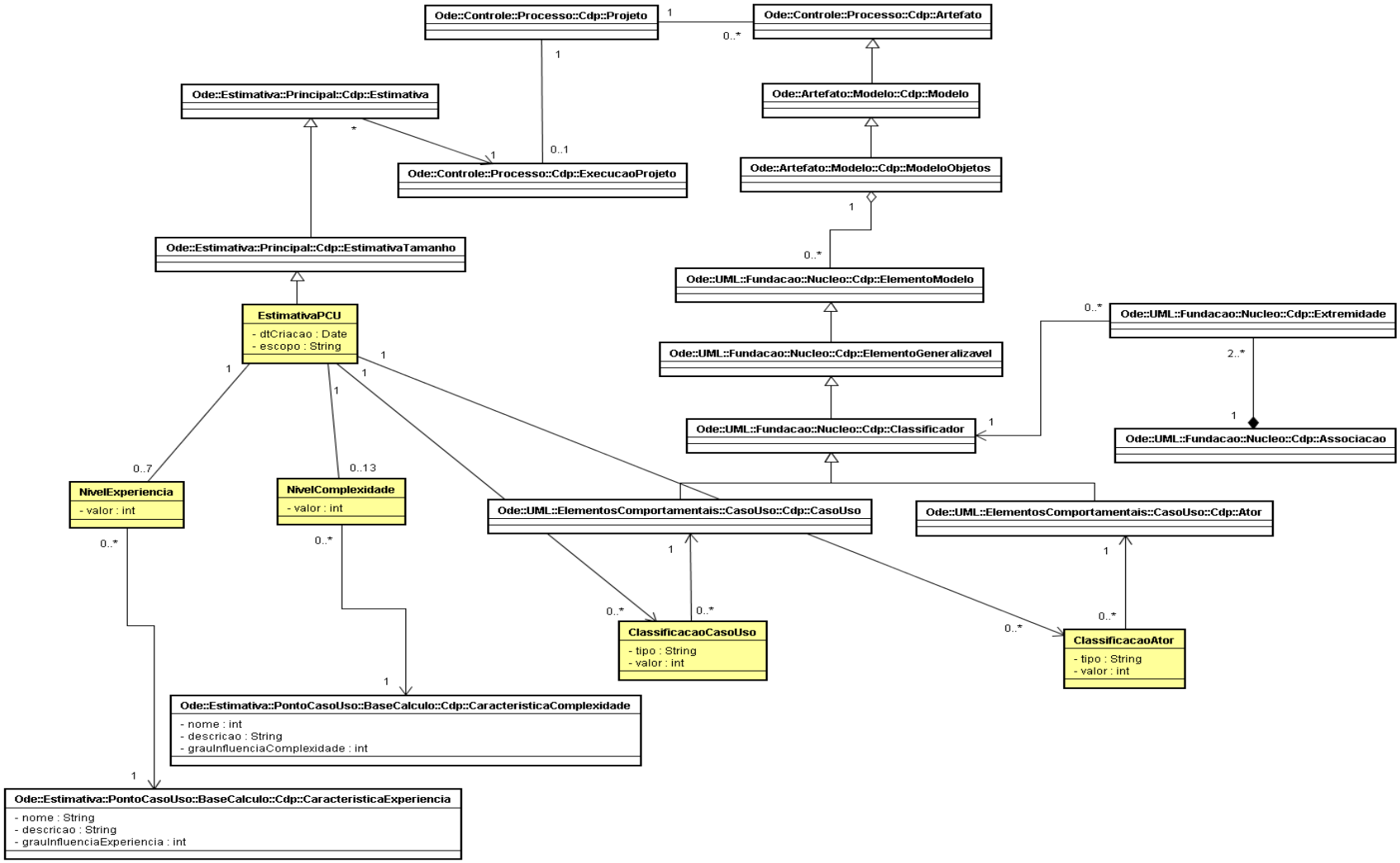

Figura B.7 – Componente de Domínio do Problema do pacote *Analise*

B.3.2.2 – Componente de Gerência de Dados (Cgd)

As classes contempladas no Cgd têm a função de abstrair o acesso ao banco de dados usado na implementação do trabalho. A Figura B.8 mostra o diagrama de classes desse pacote.

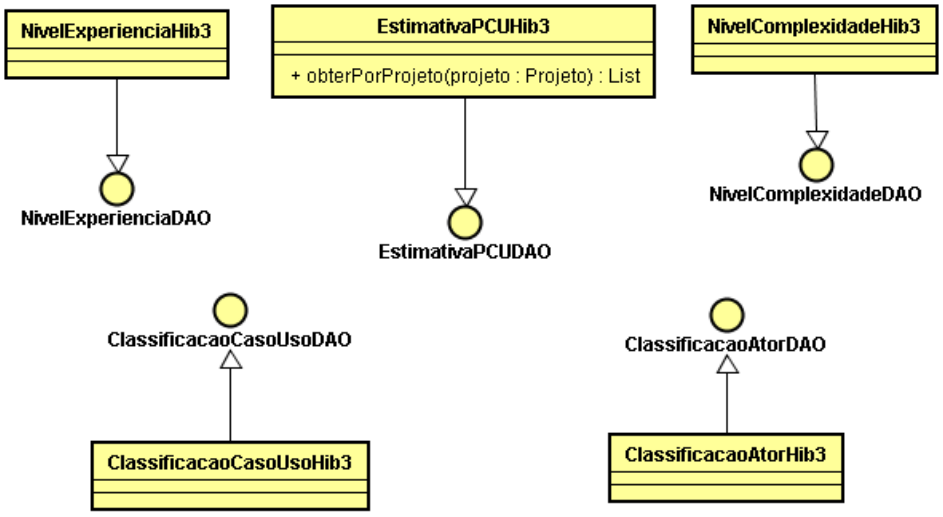

Figura B.8- Componente de Gerência de Dados do pacote *EstimativaPCU*

Todas as classes mostradas na Figura B.8 herdam da classe abstrata `Hibernate3DAO` que já implementa métodos básicos de manipulação de objetos, tais como *salvar*, *excluir*, *obterPorId* e *obterTodos*.

B.3.2.3 – Componente de Gerência de Tarefas (Cgt)

O componente de Gerência de Tarefas desse pacote possui uma única classe, que é responsável por realizar todos os casos de uso definidos no documento de requisitos, como mostra a Figura B.9.

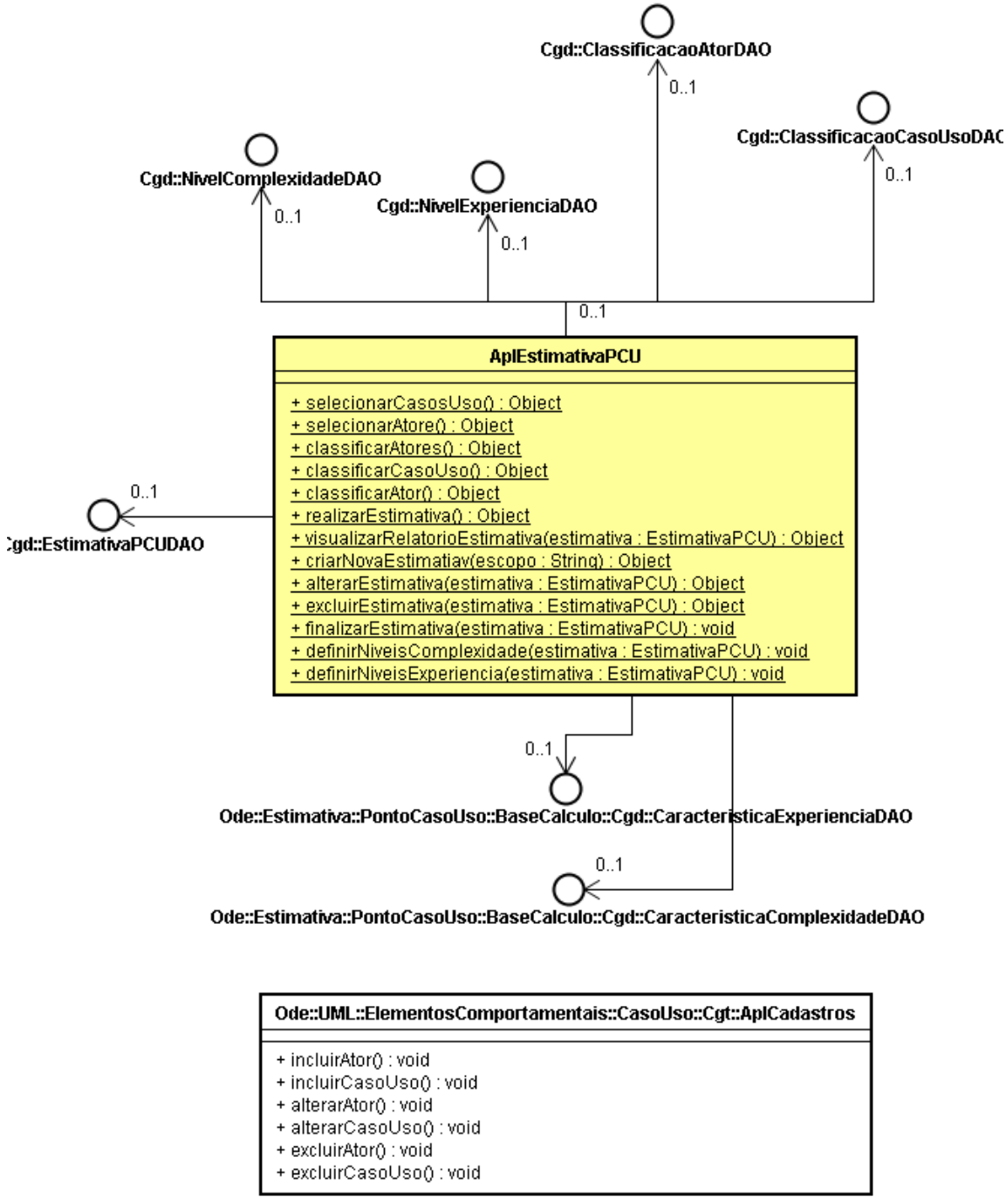

Figura B.9- Componente de Gerência de Tarefas do Pacote *EstimativaPCU*

O evento *Copiar Estimativa* do caso de uso *Controlar Estimativa* não foi contemplado nesta versão da ferramenta, portanto não há operação relativa a esse evento no modelo da Figura B.9. A operação *selecionarAtores* foi necessária, visto que ainda não foi

implementada a funcionalidade de selecionar os atores automaticamente a partir de um caso de uso selecionado.

Os métodos implementados são estáticos e podem ser acessados sem a necessidade de instanciar objetos da classe de aplicação, provendo facilidade aos desenvolvedores que utilizam essas operações.

B.3.2.4 – Componente do Componente de Interação Humana (Cih)

O componente de Interação Humana do pacote *EstimativaPCU* materializa as interfaces com o usuário para a realização dos casos de uso definidos no documento de especificação de requisitos. Para a implementação desse componente foram seguidos os novos padrões de interface para ferramentas internas do ambiente ODE. A Figura B.10 mostra o diagrama de classes desse componente.

A janela de cadastro de estimativas (`JanCadastroEstimativa`) apresenta as estimativas de pontos de casos de uso cadastradas no sistema e permite que se faça a criação de uma nova estimativa, a alteração ou exclusão de uma estimativa existente e a geração de relatório de uma estimativa. `JanRealizarEstimativa` agrupa e organiza a ordem de exibição dos painéis que guiam os passos da análise de PCUs, fazendo parte de um novo padrão para o ambiente ODE. O `PainelDefinirEscopo` permite que o gerente escolha o tipo de escopo de contagem, logo fazendo parte do evento *Criar Nova Estimativa*. A seleção dos atores e casos de uso é contemplada pelas classes `PainelSelecaoAtor` e `PainelSelecaoCasoUso` respectivamente. Da mesma forma `PainelClassificarAtor` e `PainelClassificarCasoUso` permitem a classificação de atores e casos de uso de uma contagem quanto à sua complexidade.

Os painéis `PainelCadastrarAtor` e `PainelCadastrarCasoUso` foram desenvolvidos neste pacote, pois não foi possível reutilizar as respectivas interfaces da ferramenta OODE, pois nessa ferramenta, desenvolvida segundo a antiga concepção adotada em ODE de quatro camadas, há um forte acoplamento entre as classes de interface e as classes de aplicação, incompatível com a nova concepção adotada.

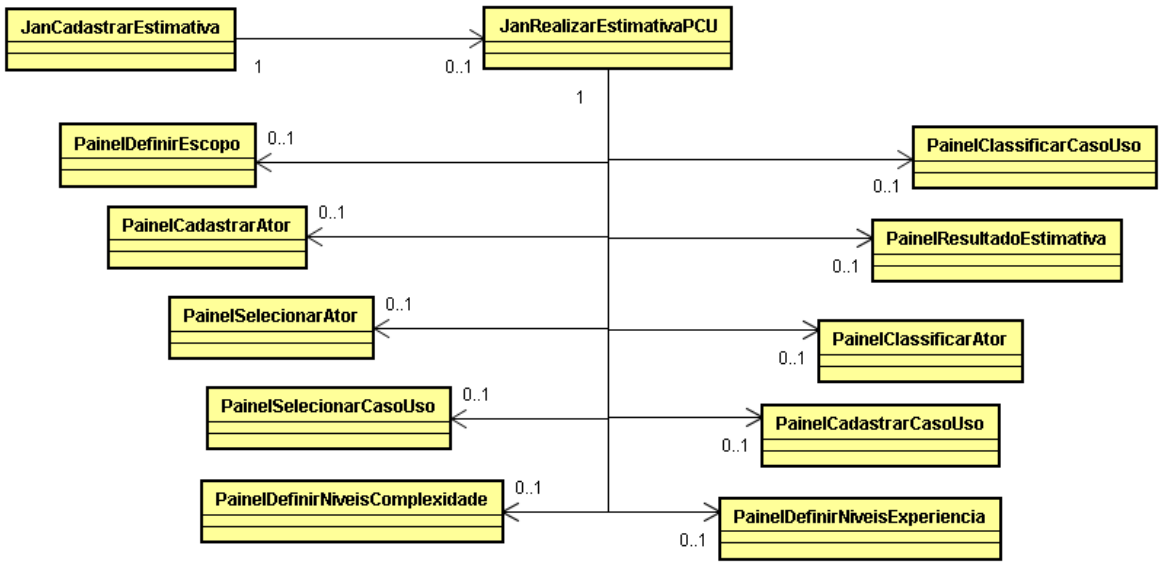

Figura B.10 - Componente de Interface Humana do pacote *EstimativaPCU*

As definições dos níveis de complexidade e de experiência são realizadas por meio das interfaces `PainelDefinirNiveisComplexidade` e `PainelDefinirNiveisExperiencia`, que exibem todos os níveis relacionados à estimativa vigente (escolhida na janela de cadastro de estimativas) permitindo que os valores dos níveis sejam alterados. Por fim o `PainelResultadoEstimativa` mostra o resumo da contagem.

A interface `PainelSelecaoAtor` foi adicionado ao modelo para suprir a falta momentânea da funcionalidade de selecionar os atores automaticamente a partir de um caso de uso selecionado.

B.3.2.5 – Componente de Controle de Interação (Cci)

As classes do pacote Cci são responsáveis por fazer o intercâmbio de dados entre as classes de aplicação e as classes de interface com o usuário da ferramenta, como mostra a Figura B.11.

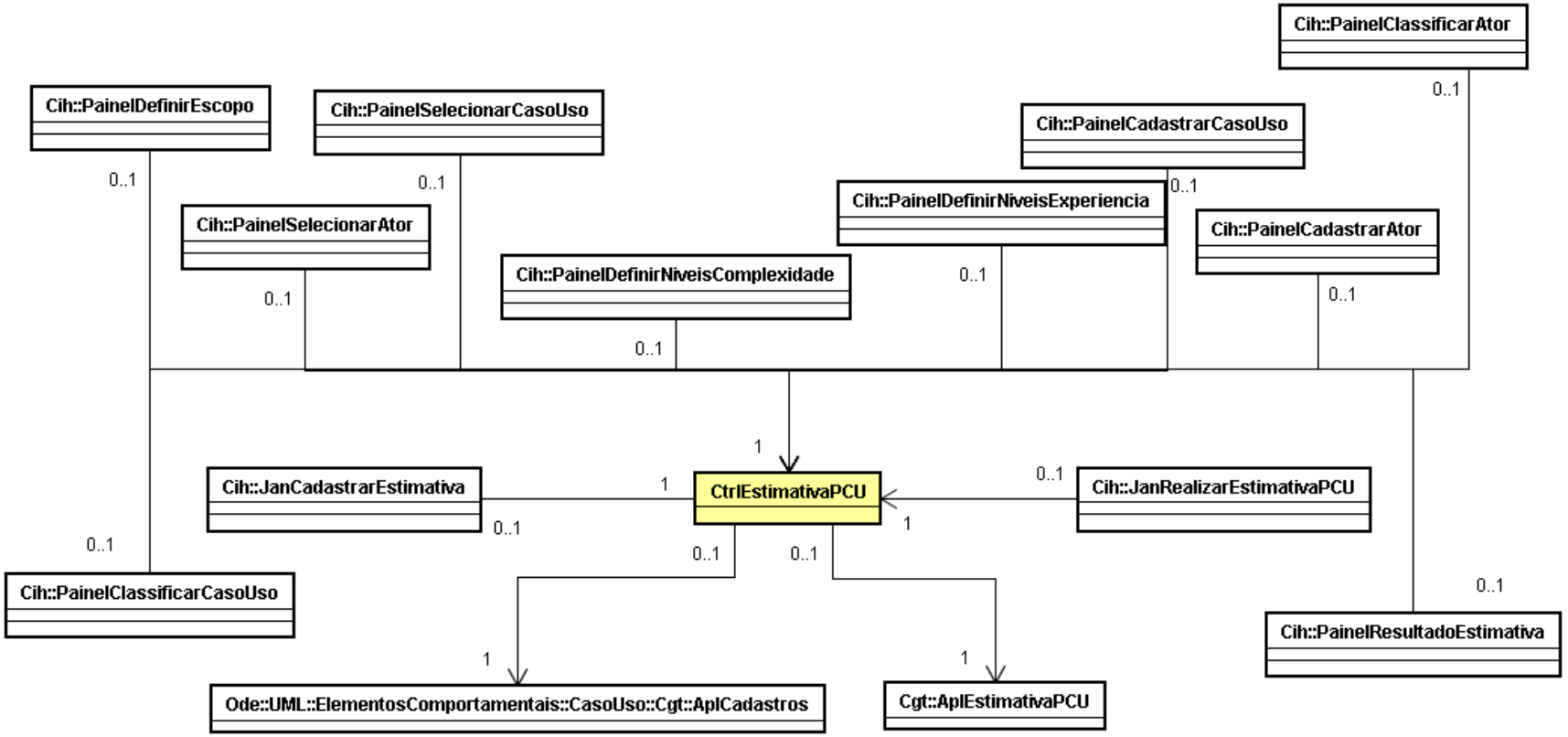

Figura B.11 - Componente de Controle de Interação do Pacote *EstimativaPCU*

A navegabilidade no sentido `CtrlEstimativaPCU` e `AplEstimativaPCU` permite que as funcionalidades da ferramenta possam ser utilizadas por outras ferramentas. Além disso, se houver a necessidade de modificar a interface da ferramenta, basta reformular o controlador de acordo com mesma.

As operações relacionadas ao cadastro de casos de uso e atores estão definidas na classe `AplCadastros` pertencente ao pacote *Ode::UML::ElementosComportamentais::CasoUso::Cgt,* sendo, portanto, um exemplo do bom funcionamento da nova arquitetura proposta.

Toda vez que uma interface necessita acessar alguma funcionalidade, provida por uma classe de aplicação (`AplEstimativaPCU` ou `AplCadastros`), ela o faz via `CtrlEstimativaPCU`, que conhece as operações necessárias para atender à requisição da interface. Quando a aplicação requisitada retorna um resultado para o controlador, esse interpreta o mesmo e atualiza as interfaces de acordo com a necessidade.

B.3.3 Pacote BaseCalculo

O pacote *BaseCalculo* contém as classes que apóiam a contagem dos pontos de caso de uso, contemplando todos os valores-padrão relativos ao método de estimativa de PCU. Como esses valores são fixos e não há funcionalidade para criar novos, alterar ou excluir, o pacote *BaseCalculo* é responsável apenas por tratar do domínio do problema e da gerência de dados.

B.3.3.1 – Componente do Domínio do Problema (Cdp)

As classes `FatorPesoAtor` e `FatorPesoCasoUso` guardam estaticamente as tabelas definidas no método, como mostra a Figura B.12. `CaracteristicaComplexidade` e `CaracteristicaExperiencia` contemplam as características de complexidade e de experiência definidas pelo método de estimativas por pontos de caso de uso.

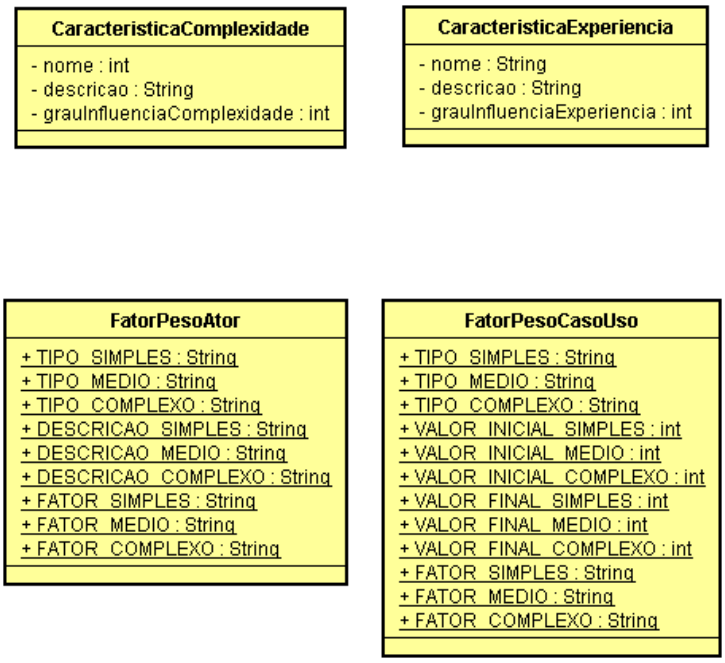

Figura B.12 - Componente de Domínio do Problema do Pacote *BaseCalculo*

B.3.3.2 – Componente de Gerência de Dados (Cgd)

Conforme citado anteriormente, as classes contempladas no Cgd têm a função de abstrair o acesso ao banco de dados usado na implementação do trabalho. A Figura B.13 mostra o diagrama de classes desse pacote..

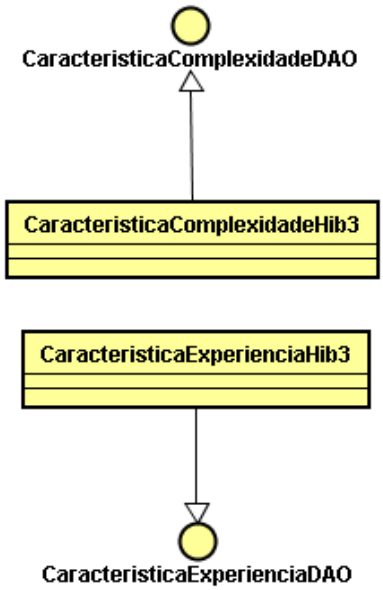

Figura B.13- Componente de Gerência de Dados do Pacote *BaseCalculo*

Como os fatores de peso foram implementados estaticamente nas classes de domínio, não há necessidade de ter classes de gerência de dados para eles. Assim, as únicas classes que devem ser tratadas nesse pacote são as mostradas na Figura B.13.

B.3.4 A Ferramenta

Serão apresentadas nesta seção, algumas interfaces dos eventos mais relevantes da ferramenta, a saber Seleção dos Casos de Uso de uma Contagem e Classificação dos Casos de Uso.

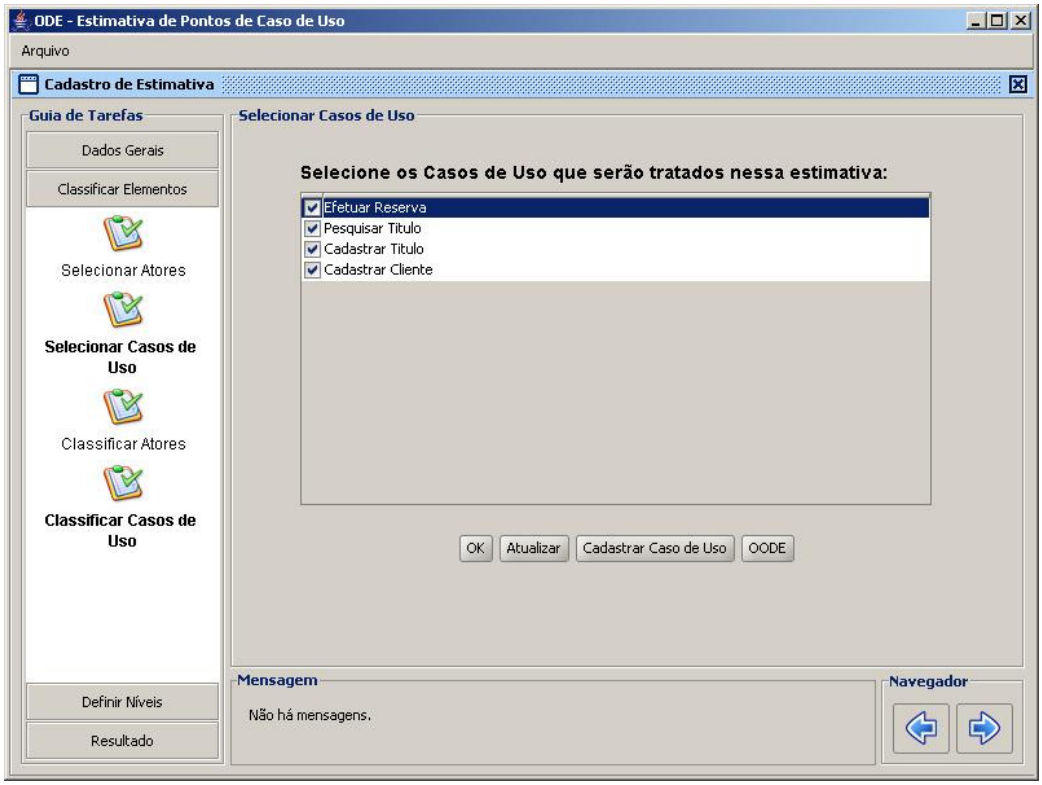

Figura B.14- Seleção dos Casos de Uso para uma contagem

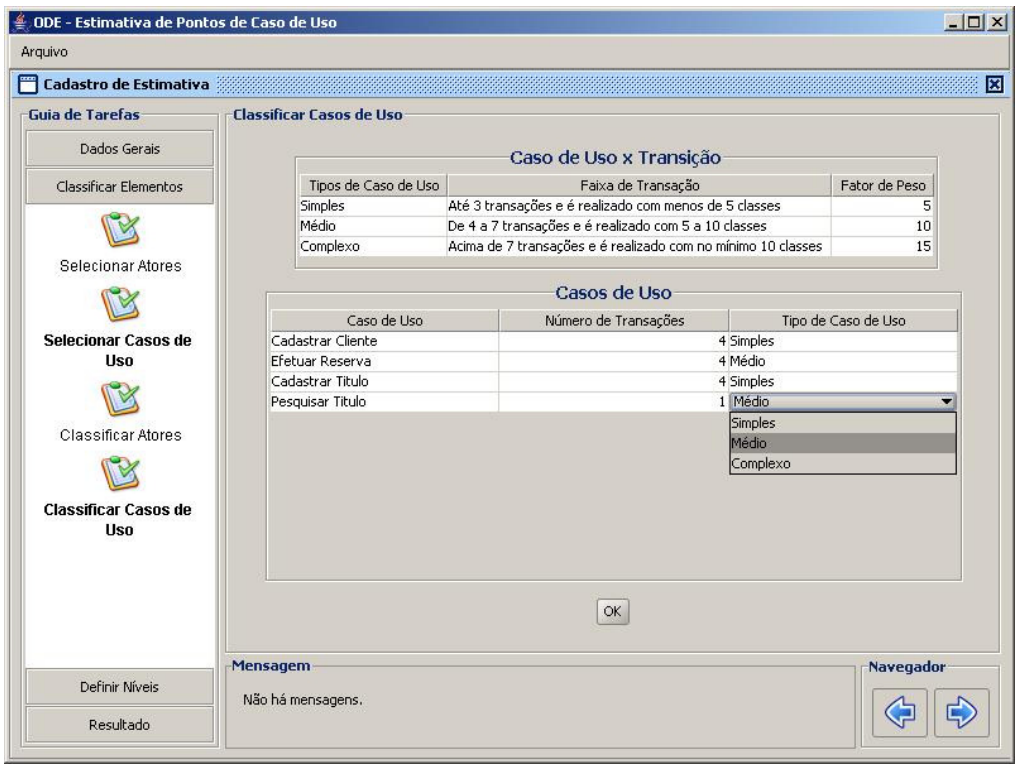

Figura B.14 – Classificação dos Casos de Uso