UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO
EM COMPUTAÇÃO APLICADA – PIPCA

ANDRÉ LOPES TOCCHETTO

AspectCost: um ambiente de gerência e acompanhamento de custos de requisitos baseados em AOP.

Dissertação apresentada ao Curso de Mestrado em Computação Aplicada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Sérgio Crespo.

São Leopoldo

2007

ANDRÉ LOPES TOCCHETTO

AspectCost: um ambiente de gerência e acompanhamento de custos de requisitos baseados em AOP.

Dissertação apresentada ao Curso de Mestrado em Computação Aplicada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Sérgio Crespo.

São Leopoldo

2007

Ficha catalográfica elaborada pela Biblioteca da
Universidade do Vale do Rio dos Sinos

T631a Tocchetto, André Lopes

AspectCost: um ambiente de gerência e acompanhamento de custos de requisitos baseados em AOP / por André Lopes Tocchetto. – 2007.
101p. : il. ; 30cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2007.
"Orientação: Prof. Dr. Sérgio Crespo C. S. Pinto, Ciências Exatas e Tecnológicas".

1. Desenvolvimento de *software*. 2. Programação orientada a aspecto. 3. Custo de requisito - Projeto. I. Título.

Catalogação na Publicação:
Bibliotecária Eliete Mari Doncato Brasil - CRB 10/1184

Dedico este trabalho as pessoas mais importantes da minha vida: Mãe, Pai, Felipe, Guilherme, Graciela e Jaque, pelo amor e apoio.

AGRADECIMENTO

Agradeço....

... à minha Mãe pelos votos de compreensão, incentivo e motivação em momentos difíceis.

... à minha Mãe por ser minha mãe.

... à meu Pai que por motivos maiores não está presente para partilhar das conquistas e dificuldades.

... aos meus irmãos, Felipe, Guilherme e Graciela que sempre estiveram ao meu lado incentivando.

... ao Professor Dr. Sérgio Crespo pela orientação, dedicação e incentivo durante a pesquisa.

... ao Professor Dr. Sérgio Crespo pela bolsa que financiou este mestrado.

... aos meus colegas, Cristiano Galina, Rogério Martins, Tiago Minuzzi, Júnior Martins, Patrícia Cavedini, Gustavo Bisognin e Cícero Rolim que foram amigos durante estes dois anos.

... a todos que de alguma forma contribuíram pelo desenvolvimento do presente trabalho.

"*Uma longa viagem começa com um único passo*".
Lao-Tsé.

RESUMO

Uma vez que o desenvolvimento e a necessidade de novos paradigmas de desenvolvimento de *software* com o objetivo de facilitar a criação de soluções para os problemas cada vez mais complexos são gradativos, e considerando o desenvolvimento de uma solução em um ciclo cascata clássico (análise, projeto, codificação, testes e manutenção), pode-se afirmar que todas as etapas devem prover um *framework* para contemplar os artefatos que fazem parte desses novos paradigmas. Sendo assim, cita-se o desenvolvimento da UML para representar um *software* que utiliza o paradigma de desenvolvimento orientado a objetos, o qual tem algumas funcionalidades que estão espalhadas durante todo o *software* dificultando a implementação e conseqüentemente a evolução. Então, com vistas a solucionar esse problema, propõem-se a utilização do paradigma de desenvolvimento orientado a aspecto para suprir a carência do paradigma orientado a objeto. Porém, há a necessidade, conforme mencionado anteriormente, de que todos os *frameworks* acompanhem essa evolução. Portando, este trabalho visa adaptar a o método de estimativa *Use Case Point* para contemplar os casos de uso aspectuais, para que esses artefatos sejam inseridos no cálculo do custo total do projeto. Assim, adicionando a utilização do processo AHP para calcular os custos dos requisitos, a metodologia apresentada contribui para a fase de análise de um *software* que utiliza o paradigma de desenvolvimento orientado a aspecto.

Palavras-Chaves: aspectos, custos, e requisitos.

ABSTRACT

Since the development and the necessity of new paradigms of software development with the objective of facilitating the creation of solutions for problems that are more and more complex are gradual, and considering the solution development in a classic cascade cycle (analysis, project, codification, tests and maintenance), it is possible to state that all the stages must provide a framework to contemplate the devices that are part of these new paradigms. Thus, the development of the UML is mentioned in order to represent a software that uses the development paradigm oriented to objects, which has some functionalities that are spread throughout the software, making it difficult for its implementation and consequently its the evolution. So, aiming to solve this problem, the usage of the development paradigm oriented to aspect is proposed in order to supply this lack of the paradigm oriented to object. However, there is the necessity, as previously mentioned, of all frameworks following this evolution. Therefore, this work aims to adapt the estimate method called Use Case Point to contemplate the aspectual use cases, so that these devices shall be inserted in the total cost calculation of the project. This way, adding the use of AHP process to calculate the requirements costs, the presented methodology contributes for the analysis phase of a software that uses the development paradigm oriented to aspect.

Key words: aspects, costs, and requirements.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHP - *Analytical Hierarchy Process*
CA - Consistência Aleatória
CAA - Complexidade do Ator Associado
CMM - *Capability Maturity Model*
CMMI - *Capability Maturity Model Integration*
COCOMO - *Constructive Cost Model*
COUC - Complexidade dos Diagramas de Caso de Uso
CP - Pontos de Classe
CUCA - Complexidade do Caso de Uso associado à classe
EF - *Environment Factor*
EMF - *Equivalent Modification Factor*
FP - *Function Points*
GUI - *Graphical User Interface*
MUAW - *Modified Unadjusted Actor Weight*
MUUCP - *Modified Unadjusted Use Case Point*
MUUCW - *Modified Unadjusted Use Case Weight*
NA - Número de classes associadas
PMI - *Project Management Institute*
RC - Valor de Consistência
RUP - *Rational Unified Process*
SLOC - *Source Lines of Code*
SPICE - *Software Process Improvement and Capability dEtermination*
TCF - *Technical Complexity Factor*
UAW - *Unadjusted Actor Weight*
UCP - *Use Case Points*
UML - *Unified Modeling Language*
UUCAW - *Unadjusted Use Case Aspect Weight*
UUCP - *Unadjusted Use Case Point*
UUCW - *Unadjusted Use Case Weight*
VIM - *Incosistency Management*
XP -*Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

A exigência de soluções mais robustas e complexas para suprir necessidades, como agilidade nos processos e gerenciamento de informações, é uma tendência crescente dentro do contexto de desenvolvimento de *software*. Sendo assim, o crescimento dos problemas exige que as soluções estejam disponíveis também para prover o controle produtivo e evolutivo das organizações que desenvolvem determinada solução.

Portanto, visando a gerenciar, controlar e satisfazer as exigências dos *stakeholders,* a gerência de projeto é um grande desafio das organizações e dos gestores nos tempos modernos. Além disso, a preocupação com o sucesso faz com que essa área seja cada vez mais aplicada, estudada, conhecida, difundida, implementada e desenvolvida, e, por isso, para reduzir as dificuldades de gerenciamento de projetos e auxiliar as organizações, os gestores, os pesquisadores e as instituições, dispõem de uma série de metodologias, dentre as quais as mais conhecidas são: CMM/CMMI, PMBOK-PMI, RUP, SPICE e XP (Wasileski, 2005).

Segundo o *Project Management Institute* (PMBOK, 2000), a gerência de projeto é a aplicação de conhecimento, habilidades e técnicas para projetar atividades que visem ou excedam às necessidades e às expectativas das partes envolvidas. Sendo que um projeto é considerado uma atividade empreendedora única, sempre delimitada com início e fim; usufruindo de recursos disponíveis, e controlado por pessoas, com objetivo de atingir metas pré-definidas, cujo sucesso tem seus parâmetros estabelecidos pela qualidade, pelos custos e pelos prazos (Karlsoon, 1997).

No entanto, dentro do contexto de desenvolvimento de *software* existem inúmeros problemas que exigem da gerência uma postura criteriosa, como a tomada de decisão e as atividades ligadas ao projeto, que estão diretamente relacionadas ao escopo. Então, com o objetivo de ter uma dimensão do problema estabelecido pelos interessados, a Engenharia de Requisitos disponibiliza técnicas para a descoberta das necessidades particulares destas (Nuseibeh, 2000), pois, cada vez mais, há a necessidade de se estimar custos antecipadamente considerando o ciclo de desenvolvimento de um *software* (Matson, 1994).

As estimativas e o controle de custos impactam diretamente no valor total de um projeto, nos prazos e na sua qualidade, sendo muito mais uma arte do que uma ciência (Mohagheghi, 2005). Assim, por serem as mais conhecidas, citam-se: *Source Lines of Code*

(SLOC), *Function Points* (FP) e *Constructive Cost Model* 81 (COCOMO); embora elas não considerem o paradigma de desenvolvimento orientado a objetos utilizando a UML como linguagem de modelagem (Mohagheghi, 2005).

A difusão da tecnologia de desenvolvimento orientado a objetos, em várias organizações, muitos pesquisadores propuseram métricas para mensurar o tamanho da complexidade dos *softwares*. Algumas métricas propostas são: funcionalidade do sistema, comunicação entre os objetos e a porcentagem de reutilização (Carbone, 2002). Segundo Pressman (Pressman, 2002), a aplicação de métricas padronizadas é o ponto de êxito para as estimativas, uma vez que direciona o processo para uma base estatística contemplando, dessa forma, os projetos anteriores. Isso significa que esta base pode ser utilizada para novos projetos e que a não utilização de métricas determinativas provoca algumas divergências entre os valores calculados em outros projetos.

Motivado pelo contexto apresentado, pela importância do gerenciamento de projetos, e pela necessidade do estabelecimento de controle de requisitos, desde fase inicial até a final, a motivação deste trabalho foi de propor o desenvolvimento de uma metodologia que contemple as citações acima descritas.

1.1 MOTIVAÇÃO

Apesar da evolução das tecnologias dos projetos e do aumento da sua complexidade, as empresas ainda adotam antigos modelos de gerenciamento, como, por exemplo, a base de conhecimento de projetos anteriores (Wasileski, 2005). Apensar de ser uma boa técnica, a evolução tecnológica permite identificar que os projetos dificilmente são similares.

De acordo com o *Standish Group*, os insucessos vêm diminuindo em decorrência do aumento da eficiência das medidas de orçamento, custos e especificações (The Standish Group, 2004).

Portanto, como o controle e a gerência dos requisitos são processos ligados ao sucesso de um projeto, a mudança de um requisito deve garantir as questões econômicas e contribuir para a regra de negócio da organização que solicita o sistema. Sendo que o controle de

mudanças consiste no estabelecimento e na execução de um processo para monitoramento, verificação e gerenciamento.

Segundo Fox, a utilização de ferramentas de gerenciamento proporciona uma mensuração prévia da estrutura e sistemática do projeto, sendo, portanto, um recurso essencial no contexto gerencial. Além disso, essas ferramentas de gerenciamento garantem aos gestores, a visualização minuciosa do andamento do projeto e a verificação de pontos errôneos (Fox, 2005).

O presente trabalho objetiva o desenvolvimento de uma metodologia para estimar o custo total de um projeto, além de mensurar e controlar os custos dos requisitos, considerando a utilização do paradigma de desenvolvimento orientado a aspectos, estendendo o modelo de estimativa *Use Case Points* (UCP).

1.2 PROBLEMA

A complexidade dos sistemas cresce constantemente e a demanda para produzir *software* torna-se um desafio, uma vez que esses sistemas necessitam de características que atendam a funcionalidades complexas. Além disso, considerando que uma mesma funcionalidade pode estar dispersa em vários módulos do sistema, a sua manutenção torna-se uma tarefa árdua.

Por conseguinte, o desenvolvimento recente do paradigma de desenvolvimento orientado a aspectos e as abordagens para gerenciar os projetos que utilizam esse paradigma determinam a necessidade de estimativas para este contexto. No entanto existem algumas estimativas que não são eficazes para os paradigmas de desenvolvimento mais atuais. Por exemplo, o uso de estimativa baseado em linha de código apresenta problemas, tais como estes que Nelson enumera (Nelson, 1999):

- Linguagens de programação variam no número de instruções para processar um comando. Por exemplo, executar um comando em Delphi requer algumas centenas de linhas de código, enquanto o mesmo comando em Java pode ser requisitado pela metade;

- Estimativas por linhas de código são baseadas em visões técnicas e físicas do sistema, porém, isso não existe nas fases iniciais de um projeto;

- Não existe um padrão definido ou método para mensurar linhas de código.

A metodologia de estimativa de análise de pontos por função não contempla os fatores de sistemas que utilizam o paradigma de desenvolvimento orientado a objeto, principalmente quando a modelagem é expressa em UML (Mohagheghi, 2005). Estes são exemplos de metodologias e métricas utilizadas atualmente que não atendem aos problemas já mencionados.

1.3 QUESTÃO DA PESQUISA

Após identificando o problema, foi estabelecida a seguinte questão de pesquisa para ser respondida nesta dissertação: como estimar, controlar e simular os custos de requisitos em projetos que utilizam o paradigma AOP?

A partir dessa questão, foram definidas algumas premissas, que são:

a) Usar estimativas de custos que considerem os conceitos de orientação a objetos (OO), visto que a orientação a aspecto complementa e não substitui a OO;

b) Utilizar uma representação gráfica e quantitativa para representar os aspectos;

c) Valer-se de uma estimativa baseada em modelos de análise com a utilização da UML, como linguagem de representação;

d) Empregar medidas quantitativas para atribuir custos aos requisitos e fatores para relacioná-los.

1.4 OBJETIVOS

Uma vez identificadas à questão de pesquisa e as premissas a elas associadas, definiu-se o seguinte objetivo para o trabalho: desenvolver uma metodologia de estimativa de custos para projetos utilizando o paradigma de desenvolvimento orientado a aspecto, visando a mensurar os custos dos requisitos e a controlá-los.

Com vistas a alcançar esse objetivo principal, definiram-se os seguintes objetivos específicos:

- Realizar um estudo sobre técnicas de estimativas de custos;
- Realizar um estudo sobre controle de custos em requisitos;
- Modelar os recursos a serem desenvolvidos;
- Implementar uma ferramenta para apoiar a metodologia;
- Validar a metodologia proposta em sistemas que utilizam o paradigma de desenvolvimento orientado a aspectos;
- Tornar a proposta adaptável a modelos completos de estimativa de recursos, como o COCOMO II.

1.5 ORGANIZAÇÃO DO VOLUME

A estruturação das seções deste volume visa a oferecer uma base inicial sobre gerência de projetos, e, por isso, é focada na estimativa de custos, no modelo de estimativa baseado em casos de uso, e culmina com a apresentação final da proposta de trabalho.

Assim, sua estrutura é a seguinte:

- Capítulo 2: Revisão bibliográfica - descreve os conceitos e tecnologias utilizadas no desenvolvimento do trabalho, abordando a gerência de projetos, a estimativa baseada

em casos de uso, o modelo RUP, o processo quantitativo AHP, e a orientação a aspectos;

- Capítulo 3: Trabalhos relacionados - apresenta algumas das abordagens que utilizam estimativa baseada no paradigma de desenvolvimento orientado a objetos, assim como também a comparação entre elas e a proposta apresentada;

- Capítulo 4: AspectCost – descreve o trabalho propriamente dito, com suas características e o detalhamento da metodologia, considerando cada etapa;

- Capítulo 5: AspectCostTool – mostra a ferramenta que apóia toda a metodologia apresentada, e ainda, descreve a solução desenvolvida e aplicada no estudo de caso;

- Capítulo 6: Estudo de caso - apresenta a utilização da solução apresentada e a análise dos dados gerados;

- Capítulo 7: Conclusões - apresenta as considerações finais quanto ao desenvolvimento do trabalho, indicando seus pontos de contribuição e os trabalhos futuros que podem ser feitos junto à ferramenta.

2. REVISÃO BIBLIOGRÁFICA

Esta seção tem por objetivo contextualizar e apresentar sucintamente todas as áreas e artefatos que servirão de base para o desenvolvimento do trabalho.

2.1 GERÊNCIA DE PROJETO

Para obter o sucesso em um projeto, além do entendimento claro das necessidades dos interessados, o gerenciamento é fundamental, pois o ato de gerenciar pode ser entendido e conduzido de várias formas, dependendo exclusivamente da cultura da empresa e dos seus objetivos. Além disso, o fator de insucesso de um projeto relaciona-se com a falta de controle, sendo que uma das alternativas para obter esse controle é a observação constante das variações do plano do projeto (Krasna, 1998).

O ato de gerenciar um projeto deve ser feito ao longo de todo o ciclo de vida de um *software*, sendo praticado em cada etapa do processo, pois o ciclo de vida de um projeto está diretamente ligado ao trabalho técnico que deverá ser desenvolvido e aos papéis envolvidos nessa fase (Martin, 2005).

Assim, dentre as atividades da gerência de projeto destacam-se as seguintes (PMBOK, 2000):

- Planejamento: é criado a partir dos objetivos e do escopo do projeto. Portanto, todas as soluções alternativas devem ser consideradas, bem como, a identificação das restrições administrativas e técnicas, uma vez que essas informações possibilitam a elaboração de uma estimativa de prazos, recursos, esforços, custos e tamanho do projeto. Além disso, o processo de desenvolvimento a ser adotado, a estrutura da organização e os pontos de controle são definidos a partir do planejamento;

- Controle: após feita a avaliação do escopo do projeto, é possível fazer a avaliação e a identificação dos recursos pessoais e das ferramentas necessárias para a execução do mesmo, pois é através dela que se identificam as

habilidades necessárias para a conclusão do desenvolvimento, os recursos necessários, a duração das tarefas, as atividades a serem feitas, e os períodos requeridos para as ferramentas de apoio. Além disso, o gerente do projeto é responsável pelo acompanhamento das tarefas e das atividades estabelecidas, assim como os recursos definidos no planejamento podem ser redirecionados, as tarefas realocadas, e os compromissos de entregas modificados, para suprir um possível problema não coberto ao longo do projeto. Sendo assim, o controle é feito através do controle de métricas e dados quantitativos;

- Monitoramento: todo o progresso é rastreado pelo gerente, que avalia a necessidade de alocar novos recursos. Além disso, a avaliação da produtividade da equipe é um dos fatores mais relevante para o sucesso, pois impacta nos prazos, nos custos e na qualidade do projeto.

Também é importante salientar que o PMI, através do PMBOK, divide a gerência de projeto em nove áreas, as quais são (PMBOK, 2000):

- **Gerência de Custos:** processo necessário para garantir que o projeto seja concluído dentro do orçamento previsto; o qual é composto pelo planejamento de recursos e estimativa, orçamento, e controle de custos;

- **Gerência de Tempo:** processo necessário para garantir a conclusão dentro do prazo previsto; que é composto pela definição, estimativa de duração e seqüência das atividades, além do cronograma de desenvolvimento e controle do mesmo;

- **Gerência de Escopo:** processo necessário para garantir a conclusão do trabalho requerido; composto pela iniciação, planejamento de escopo, detalhamento do escopo, verificação do escopo, e controle de mudanças do escopo;

- **Gerência da Qualidade:** processo necessário para garantir que as necessidades do cliente serão satisfeitas; o qual é composto pelo planejamento da qualidade, assim como seu controle;

- **Gerência de Comunicação:** processo necessário para garantir que a geração, a captura, a distribuição, o armazenamento e as apresentações do projeto serão feitos de forma adequada e no tempo certo; e que é composto pelo planejamento das comunicações, distribuição das informações, relato de desempenho, e encerramento administrativo;

- **Gerência de Recursos Humanos:** processo necessário para proporcionar uma melhor utilização dos integrantes do projeto; o qual é responsável pelo planejamento organizacional, além da montagem e do desenvolvimento da equipe;

- **Gerência de Riscos:** processo necessário relacionado à identificação, à análise e à resposta aos riscos do projeto; que é composto pela identificação dos riscos, assim como também pela quantificação, desenvolvimento e controle de respostas;

- **Gerência de Integração:** processo necessário para proporcionar que todas as partes do projeto sejam de alguma forma, coordenadas; que é composto pelo desenvolvimento e execução do plano de projeto, e também pelo controle geral de mudanças;

- **Gerência de Aquisição:** processo necessário para a aquisição de serviços fora da organização que desenvolve o projeto; o qual é composto pelo planejamento e preparação das aquisições, assim como também pela obtenção das propostas, seleção de fornecedores, além da administração e encerramento dos contratos.

Sendo assim, o foco deste trabalho, considerando as nove áreas citadas acima, concentra-se na gerência de custos, mais especificamente na estimativa e no controle, os quais serão apresentados mais detalhadamente a seguir.

2.1.1 Estimativa e controle de custos

Segundo o PMBOK, a área de gerência de custo é dividida em quatro subáreas (PMBOK, 2000):

- Planejamento de recursos: atividade que determina os recursos e a quantidade necessária de cada um deles para a execução das atividades do projeto;
- Estimativa de custos: atividade que prevê os custos dos recursos necessários para a execução das atividades relacionadas ao projeto;
- Orçamento dos custos: é a subárea em que se alocam as estimativas de custos para cada item do projeto;
- Controle dos custos: atividade que controla as mudanças no orçamento do projeto.

Estimando-se um projeto nas fases iniciais tem-se a garantia de acurácia e a credibilidade da previsão ao longo de todo o ciclo de vida. Assim, segundo Agarwal (Agarwal, 2001), o processo de estimativa consiste dos seguintes passos e/ou procedimentos:

- Tamanho;
- Custo e esforço;
- Tempo das tarefas;
- Recursos computacionais críticos.

Outros passos ainda compõem o processo:

- Análise de risco;
- Vistoriar e aprovação;
- Rastreamento;
- Medição e melhoria do processo.

É importante ainda observar que o primeiro passo é a estimativa do tamanho do projeto, ou seja, quanto de esforço é necessário para o seu desenvolvimento. Então, considerando os modelos utilizados destacam-se os heurísticos e paramétricos. Os modelos paramétricos utilizam métricas nas fases iniciais do ciclo de vida de um *software* (Agarwal, 2001), dentre as quais as mais utilizadas são: linhas de código, análise de pontos por função, casos de uso, e *Graphical User Interfaces* (GUI). Porém, não existe nenhuma métrica comum para todos os tipos de projeto, ficando a critério do gerente a escolha de sua utilização, de acordo com as particularidades de cada projeto (Agarwal, 2001).

Considerado a estimativa paramétrica citam-se o COCOMO e COCOMO II desenvolvido por Boehm (Boehm, 2000). COCOMO é um método que visa medir esforços, prazos, recursos e custos necessários para o desenvolvimento de um *software,* desde que se tenha o tamanho do mesmo, o que poder ser obtido através de um modelo de estimativa, como Pontos por Função. Na versão COCOMO II acoplam-se a utilização de heurísticas mais atuais de estimativa de tamanho, tais como número de casos de uso e pontos por objeto, ou seja, adapta-se a projetos desenvolvidos no paradigma orientado a objetos (Boehm, 2000); além de oferecer suporte para o desenvolvimento, utilizando o *framework* RUP. A utilização destas duas metodologias permite calcular o esforço necessário para cada interação realizada. Sendo possível estimar de uma forma mais precisa os fatores necessários para o desenvolvimento de um *software* (Boehm, 2000).

Os métodos heurísticos são divididos em abordagens *top down* e *botton up*; sendo que a abordagem *top down* consiste na estimativa baseada em dados históricos, como por exemplo, em custos dos projetos de análise; e, por sua vez, a *botton up* estima o esforço requerido para cada módulo da aplicação.

Através desse levantamento é possível observar que o modelo COCOMO II utiliza outras metodologias para complementá-lo; e que, portanto, a metodologia apresentada neste trabalho se adapta às novas características desse modelo.

2.2 *USE CASE POINTS*

Segundo Booch, os casos de uso especificam o comportamento de um sistema ou de parte de um sistema, e são uma descrição de um conjunto de ações seqüenciais, incluindo variantes realizadas pelo sistema para produzir um resultado observável a um determinado ator (Booch, 2000).

Além disso, o método de estimativa por casos de uso é uma extensão dos métodos de Análise por Ponto de Função e Análise de Pontos de Função MK II (Symons, 1991); e cujos elementos que fazem parte são: os atores que interagem com o sistema, os casos de uso, e os fatores de ajustes. Por sua vez, os fatores de ajustes são: os técnicos (*TCF – Technical Complexity Factor),* cujo objetivo é cobrir uma série de requisitos funcionais do aplicativo; e os fatores de ambiente (*EF –Environment Factor)*, que são os requisitos não funcionais associados ao processo de desenvolvimento.

Deve-se salientar que o *Use Case Point* é o produto de três fatores, os quais são:

- *Unadjusted Use Case Point* (UUCP);
- Fatores técnicos;
- Fatores de ambiente.

O método estima o esforço de desenvolvimento de um sistema mapeado em pessoas/horas, contemplando fases ou o projeto como um todo, sendo que o modelo de estimativa UCP pode auxiliar o modelo de estimativa COCOMO (Karner, 1993).

2.2.1 UUCP – *Unadjusted Use Case Point*

Para calcular o UUCP é necessário avaliar todos os atores que fazem parte do diagrama de caso de uso, como o simples, o médio e o complexo. Sendo assim, na Tabela 1 apresentam-se os critérios para avaliar cada um dos atores:

Tabela 1 : Critérios de avaliação dos atores

Complexidade	Definição	Peso
Simples	O ator é dito simples quando representa um outro sistema.	1
Médio	O ator é dito médio se: • A interação com outro sistema se dá através de um protocolo; • A interação humana se dá através de um terminal.	2
Complexo	O ator é dito complexo se a interação é feita através de uma interface gráfica (GUI).	3

Através da atribuição de valores a cada um dos atores dos casos de uso é possível estimar o *Unadjusted Actor Weight* conforme a equação 1:

$$\text{UAW} = \sum(\#\,Atores * Peso) \quad (1)$$

A avaliação dos casos de uso que fazem parte do diagrama e a atribuição dos pesos para cada um são relacionadas na Tabela 2:

Tabela 2 : Critério para avaliação dos casos de uso

Complexidade	Definição	Peso
Simples	O caso de uso possui até três transações (cenários) incluindo fluxos alternativos.	5
Médio	O caso de uso possui de quatro a sete transações (cenários) incluindo fluxos alternativos.	10
Complexo	O caso de uso possui mais de sete transações (cenários) incluindo fluxos alternativos.	15

Após a atribuição de valores para cada um dos casos de uso do diagrama obtém-se o *Unadjusted Use Case Weight* a partir da equação (2):

$$\text{UUCW} = \sum (\#Casosdeuso * Peso) \quad (2)$$

Por conseguinte, a soma dos dois valores representada pela equação 3 resulta no *Unadjusted Use Case Point*:

$$\text{UUCP} = \text{UAW} + \text{UUCW} \quad (3)$$

Finalmente, o próximo passo para o cálculo do *Use Case Point* é a atribuição de pesos para os fatores técnicos.

2.2.2 Fatores técnicos (TFC)

A atribuição dos valores para os fatores técnicos está diretamente relacionada às dificuldades de construção do sistema. Segundo Karner, os fatores técnicos são semelhantes aos do Pontos por Função, porém, baseiam-se em conceitos de desenvolvimento orientado a objetos (Karner, 1993). Symons define fatores técnicos como (Symons, 1993):

"Requisitos do sistema que afetam o tamanho da tarefa, mas desconsiderados no ambiente do projeto".

Karner propõe os fatores técnicos e os pesos correspondentes a cada um deles, conforme pode-se observar na Tabela 3 (Karner, 1993):

Tabela 3 : Fatores técnicos

Fator	Requisito	Peso
T1	Sistema distribuído	2
T2	Tempo de resposta	2
T3	Eficiência	1
T4	Processamento complexo	1
T5	Código reutilizável	0.5
T6	Facilidade de instalação	0.5
T7	Facilidade de uso	0.5
T8	Portabilidade	2
T9	Facilidade de mudança	1
T10	Concorrência	1
T11	Recursos de segurança	1
T12	Acessível por terceiros	1
T13	Requer treinamento especial	1

Pode-se observar que os pesos dos fatores de ambiente variam numa escala de 0 a 5, em que o zero representa o menos relevante, e o cinco, o mais essencial. Portanto, se para todos os fatores for atribuído peso 3 o TFC = 1, sendo que o TFC é obtido pela expressão (4):

$$TCF = 0.6 + (0.01 \text{ x } \sum_{1}^{13} Ti \text{ }) \quad (4)$$

Cabe ressaltar que os fatores de ambiente também afetam diretamente o cálculo.

2.2.3 Fatores de ambiente (EF)

Os fatores de ambiente indicam o quanto o sistema é eficiente, e, segundo Karner, são semelhantes aos fatores técnicos (Karner, 1993). Assim, o autor destaca oito fatores, de A1 a A8, atribuindo-lhes pesos, conforme pode-se ver na Tabela 4:

Tabela 4 : Fatores de ambiente

Fator	Requisito	Peso
A1	Familiaridade com o processo formal de desenvolvimento.	1.5
A2	Experiência com a aplicação em desenvolvimento.	0.5
A3	Experiência em OO.	1
A4	Presença de um analista experiente na equipe de projeto.	0.5
A5	Motivação da equipe envolvida.	1
A6	Requisitos estáveis.	2
A7	Desenvolvedores em meio-expediente.	-1
A8	Linguagem de programação difícil.	-1

Os pesos para os fatores de ambiente também variam em uma escala de 0 a 5, semelhante aos fatores técnicos. Assim, se para todos os fatores for atribuído o peso 4, EF equivale a 1, sendo que o valor do EF é obtido através da expressão (5):

$$EF = 1.4 + (-0.03 \times \sum_{i=1}^{8} A_i) \quad (5)$$

Após o cálculo do UUCP, dos fatores técnicos, e dos fatores de ambiente, estima-se o custo total do projeto, que é obtido a partir do produto destes três elementos, como pode ser visto na expressão (6):

$$UCP = UUCP * TCF * EF \quad (6)$$

O resultado obtido é o número de pessoas/horas necessárias para o desenvolvimento do projeto. A grande vantagem na utilização deste método é a sua simplicidade e a padronização de resultados.

Por sua vez, a metodologia de desenvolvimento RUP, que é apresentada a seguir, permite controlar os custos em fases, e fornece alguns subsídios importantes para o contexto do presente trabalho.

2.3 RUP

O *Rational Unified Process* é um processo de Engenharia de *Software*, elaborado pela Rational (Rational Unified Process, 2000), oferecendo uma orientação através da definição de tarefas e responsabilidades para o processo de desenvolvimento de *software*. Sendo seu principal objetivo garantir a qualidade do *software* produzido.

Desta forma, o *framework* proposto pelo RUP pode ser estendido e adaptado para atender às necessidades de uma determinada organização, e possui duas estruturas e/ou duas dimensões representadas pelos eixos, que podem ser visualizadas na Figura 1.

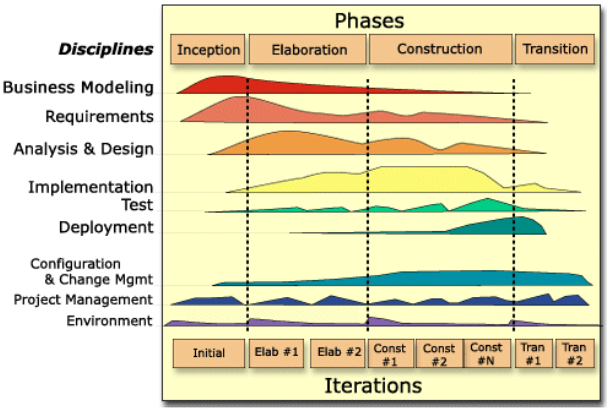

Figura 1 : Fases do RUP

Na Figura 1, o eixo horizontal representa o tempo com os aspectos dinâmicos do ciclo de vida do processo, os quais são expressos em ciclos, fases, interações e marcos. O eixo vertical representa o centro do processo, e contém cada atividade do grupo, os aspectos estáticos e descreve o conjunto de componentes, atividades, *workflows*, e artefatos.

É importante ainda salientar que as características mais relevantes desse processo são:

- Ser interativo: uma vez que tem por objetivo prover um gradativo entendimento do problema a ser resolvido através do incremento da solução durante todo o ciclo de vida (*Rational Unified Process*, 2000);

- Estar centrado na arquitetura: como o processo permite e instrui o desenvolvimento o quão antes possível do início do projeto;

- Ser dirigido por casos de uso: uma vez que o processo sugere um fluxo de ações para a realização dos casos de uso do projeto.

O *framework* RUP trabalha com quatro fases de desenvolvimento. Cada fase de desenvolvimento pode conter várias interações. As quatro fases do RUP são:

- Concepção: defini-se o escopo do projeto, sendo que para essa definição utilizam-se diagramas de casos de uso para proceder estimativas de prazos e custos;

- Elaboração: refina-se o domínio do problema e propõe-se uma arquitetura de fundação sólida, sendo assim possível eliminar elementos de maior risco para o projeto;

- Construção: desenvolve-se o produto de maneira interativa e incremental, com o objetivo de prepará-lo para a transição da comunidade usuária;

- Transição: disponibiliza-se o produto para o uso, sendo que dentro de cada fase existe um conjunto de interações que envolve planejamento, levantamento de requisitos, análise, projeto, e teste.

Portanto, a utilização do RUP neste trabalho atende a fase de divisão de valores entre as fases de um determinado projeto. Sendo que o próprio *framework* já fornece as devidas porcentagens, como pode-se observar na Tabela 5.

Tabela 5 : Divisão de esforço

	Concepção	Elaboração	Construção	Transição
Esforço	5%	20%	65%	10%

2.3.1 Desenvolvimento interativo de software

O RUP exerce um controle sobre todas as interações do projeto, as quais são planejadas e controladas, considerando a duração de cada uma e seus objetivos. Dessa forma, para cada papel as tarefas e responsabilidades são definidas, assim como, também cada objetivo é medido e avaliado.

2.3.2 Gerenciamento de requisitos

O gerenciamento de requisitos tem como objetivos descobrir, organizar, comunicar e administrar o impacto das mudanças dos requisitos de um *software*. Portanto, os benefícios mais relevantes do gerenciamento dos requisitos são:

a) Maior controle em projetos complexos;

b) Melhoria na qualidade do *software* e satisfação do cliente;

c) Redução dos custos de projeto e atrasos;

d) Melhoria na comunicação do time (Booch, 2000).

Conclui-se assim que o RUP oferece um fluxo que tem por premissa básica definir todas as funcionalidades do *software,* identificando e documentando os requisitos; e cujo objetivo é gerenciar as mudanças de requisitos, ou seja, as alterações no escopo dos mesmos. Portanto, assim o RUP provê o fluxo de configuração e gerência de mudanças.

A metodologia proposta nesta dissertação apóia a fase de gerenciamento de requisitos do RUP.

2.4 AHP – *ANALYTICAL HIERARCHY PROCESS*

O processo AHP foi criado por Saaty em 1991 com o objetivo de quantificar características qualitativas e permitindo, assim, a ponderação e a priorização de cada um dos requisitos nesse contexto. O processo utiliza a atribuição de pesos aos fatores individuais, do menos influente ao mais influente. O AHP usa de uma matriz quadrada que permite avaliar a importância de uma característica sobre a outra, a partir do cálculo do auto-vetor, obtêm-se a ordem de prioridade.

O processo AHP permite obter o valor em percentual de cada um dos requisitos e, assim, mapear o valor das fases do projeto, sendo que desta forma tem-se o valor real do custo de cada um deles.

2.4.1 O processo

A base do método é a comparação entre as características, duas a duas, a fim de avaliar a importância de cada uma utilizando-se uma escala proposta pelo autor do processo, a qual é apresentada na Tabela 6.

Tabela 6 : Escala de intensidade da importância

Intensidade de importância	Definição	Descrição
1	Menos importante.	Duas atividades contribuem igualmente para o objetivo.
3	Importância pequena.	A experiência e o julgamento favorecem levemente uma atividade em relação à outra.
5	Importância grande ou essencial.	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra.
7	Importância muito grande.	Uma atividade é fortemente favorecida. Sua dominação de importância é demonstrada na prática.
9	Importância absoluta.	A evidência favorece uma atividade em relação à outra com o mais alto grau de certeza.
2, 4, 6, 8	Valores intermediários. Se na atividade "j" recebe um dos valores acima, quando comparada com a atividade "j", então "j" tem o mesmo valor recíproco de "i".	Quando se deseja maior compromisso. É uma designação razoável.
Racionais	Razão da escala.	Se a consistência tiver de ser forçada para obter "n" valores numéricos para completar a matriz.

Para exemplificar o processo, vamos analisar o seguinte cenário:

a) Definição dos requisitos:

- Requisito 1;
- Requisito 2;

- Requisito 3;
- Requisito 4.

b) Desenho da matriz quadrada demonstrando a relação entre todos os requisitos, conforme pode ser observado na Tabela 7.

Tabela 7 : Matriz de importância entre requisitos

	Requisito 1	Requisito 2	Requisito 3	Requisito n
Requisito 1	1	a_{12}	a_{13}	a_{1n}
Requisito 2	$a_{21} = 1/a_{12}$	1	$a_{23} = 1/a_{32}$	a_{2n}
Requisito 3	$a_{31} = 1/a_{13}$	a_{32}	1	$a3n$
Requisito n	$a_{n1} = 1/a_{1n}$	$a_{n2} = a_{2n}$	$a_{n3} = a_{3n}$	1

Sendo assim, as premissas básicas para inserir os valores são:

- Se $a_{ij} = \alpha$, então $a_{ji} = 1/\alpha$, $\alpha \neq 0$;
- Se $\text{Re}\,quisito_i$ é julgado com igual importância ao $\text{Re}\,quisito_j$, então $a_{ij} = 1$, $a_{ji} = 1$ e $a_{ii} = 1$.

Portanto, atribuindo os valores para cada par dos quatro requisitos, de acordo com a escala proposta pelo autor do processo e apresentada na Tabela 6, na diagonal insere-se o valor 1. Então, considerando uma matriz de ordem n, necessita-se de n.(n-1)/2 comparações, sendo que, para o exemplo em questão, atribuíram-se os valores relacionados na Tabela 8.

Tabela 8 : Exemplo de matriz de importância

	Requisito 1	Requisito 2	Requisito 3	Requisito 4
Requisito 1	1	1/3	2	4
Requisito 2	3	1	5	3
Requisito 3	1/2	1/5	1	1/3
Requisito 4	1/4	1/3	3	1

Por sua vez, o cálculo do auto-vetor é obtido a partir da expressão (7):

$$W_i = (\prod_{j=1}^{n} a_{ij})^{1/n} \quad (7)$$

Para que os valores dos elementos fiquem na mesma unidade, normaliza-se o auto-vetor de acordo com a expressão (8):

$$T = | W_1 / \sum W_i \qquad W_2 / \sum W_i \qquad ... \qquad W_n / \sum W_n | \quad (8)$$

Para exemplificar o cálculo de T, utiliza-se como exemplo o Requisito 1:

- Somatório da Coluna do Requisito 1: T = 1 + 3 + 0,5 + 0,25 = 4,75
- Primeira Linha da Primeira Coluna normalizada: (1 * 100)/4,75 = 21,05 / 100 = 0,21
- Segunda Linha da Primeira Coluna normalizada: (3 * 100)/4,75 = 63,15 / 100 = 0,63
- Terceira Linha da Primeira Coluna normalizada: (0,5 * 100)/4,75 = 10,52 / 100 = 0,11
- Quarta Linha da Primeira Coluna normalizada: (0,25 * 100)/4,75 = 5,26 / 100 = 0,05

Esta normalização deve ser executada para n requisitos. Além disso, já que T é o valor normalizado, a sua utilização serve para quantificar e ponderar a importância das várias características de um requisito.

É importante ainda salientar que a Tabela 9 normalizada resulta na chamada matriz de prioridade.

Tabela 9 : Normalização dos valores

	Requisito 1	Requisito 2	Requisito 3	Requisito 4	Soma
Requisito 1	0.21	0.18	0.18	0.48	1.05
Requisito 2	0.63	0.54	0.45	0.36	1.98
Requisito 3	0.11	0.11	0.09	0.04	0.34
Requisito 4	0.05	0.18	0.27	0.18	0.62

Portanto,

$$\frac{1}{4} * \begin{pmatrix} 1.05 \\ 1.98 \\ 0.34 \\ 0.62 \end{pmatrix} = \begin{pmatrix} 0.26 \\ 0.50 \\ 0.09 \\ 0.16 \end{pmatrix}$$

c) Resultados

- O requisito 1 representa 26% do total do valor;
- O requisito 2 representa 50% do total do valor;
- O requisito 3 representa 9% do total do valor;
- O requisito 4 representa 16% do total do valor.

Por sua vez, a Figura 2 representa graficamente o resultado obtido para cada um dos requisitos.

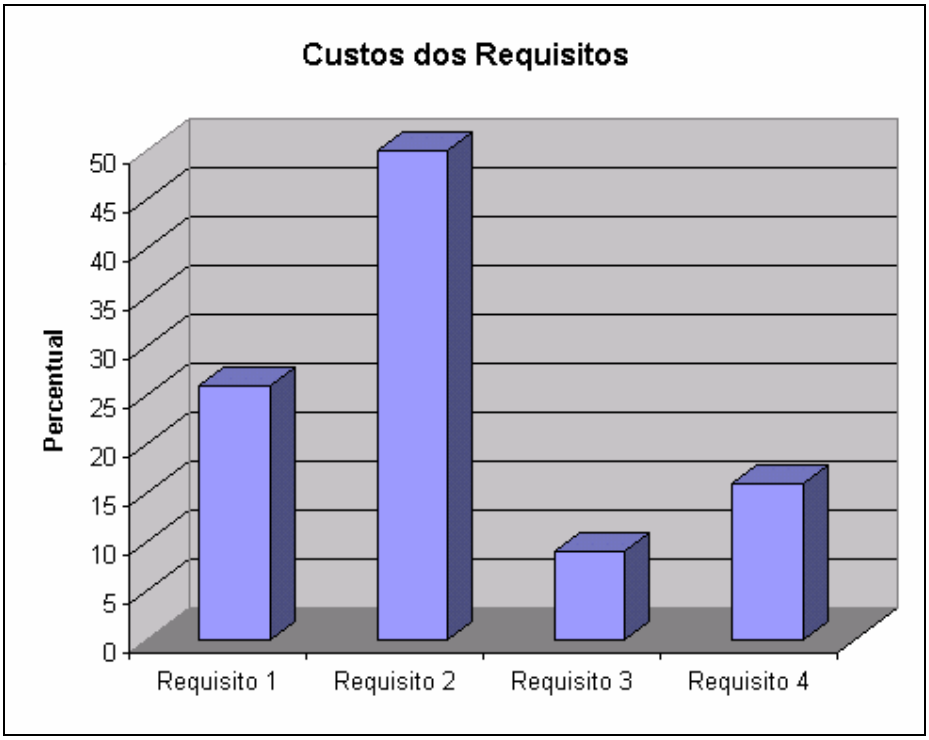

Figura 2 : Custos dos requisitos

Para verificar a consistência das respostas, o autor propõe os seguintes procedimentos:

d) Cálculo da consistência das respostas, de acordo com os procedimentos propostos pelo autor do processo (Saaty, 1991).

d1) Estimar o auto-valor (λ_{max}). A estimativa é feita através da equação (9):

$$\lambda_{max} = T.w \quad (9)$$

w = soma das colunas da matriz de comparação.

Sendo

$$\begin{pmatrix} 1 & 1/3 & 2 & 4 \\ 3 & 1 & 5 & 3 \\ 1/2 & 1/5 & 1 & 1/3 \\ 1/4 & 1/3 & 3 & 1 \end{pmatrix} * \begin{pmatrix} 0.26 \\ 0.50 \\ 0.09 \\ 0.16 \end{pmatrix} = \begin{pmatrix} 1.22 \\ 2.18 \\ 0.37 \\ 0.64 \end{pmatrix}$$

$$\begin{pmatrix} 1.22/0.26 \\ 2.18/0.50 \\ 0.37/0.09 \\ 0.64/0.16 \end{pmatrix} = \begin{pmatrix} 4.66 \\ 4.40 \\ 4.29 \\ 4.13 \end{pmatrix}$$

$$\lambda_{max} = \frac{4.66 + 4.40 + 4.29 + 4.13}{4} = 4.37$$

d2) Cálculo do índice da consistência (IC) a partir da equação (10).

$$IC = \frac{(\lambda_{max} - n)}{(n-1)} \quad (10)$$

$$IC = \frac{4.37 - 4}{4 - 1} = 0.12$$

d3) Cálculo do valor da consistência (RC), em que RC é a razão entre o índice de consistência (IC) e o índice de consistência aleatória (CA). Por sua vez, o CA é fornecido pelo processo através de um quadro (Tabela 10) obtido a partir de dados provenientes de uma amostra de 500 matrizes recíprocas geradas aleatoriamente, de tamanho 11 por 11.

Tabela 10 : Índice de consistência aleatória

n	1	2	3	4	5	6	7
CA	0	0	0.58	0.90	1.12	1.24	1.32

Sendo assim, de acordo com a Tabela 10, a CA é de ordem 4, por tratar-se apenas de 4 requisitos, cujo valor é igual a 0.90. Diante disso, a taxa para o exemplo proposto é de 0.14, obtida como resultado na expressão (11):

$$RC = \frac{IC}{CA} = \frac{0.12}{0.90} = 0.14 \quad (11)$$

Além do mais, Saaty recomenda como aceitável uma razão de consistência menor do que 10, mas, no entanto, os valores próximos também são aproveitáveis, como no exemplo do caso apresentado.

2.5 PROGRAMAÇÃO ORIENTADA A ASPECTOS

Observando algumas carências no paradigma de desenvolvimento orientado a objeto, como o espalhamento de uma mesma funcionalidade ao longo do código, já que essa técnica entrecorta a implementação em muitas partes do código, foi proposta programação orientada a aspecto. Sendo que, neste contexto, espalhamento significa uma funcionalidade comum estar implementada em vários módulos e/ou classes de um sistema orientado a objetos. Considerando que a tecnologia de orientação a objetos ainda forneça uma separação de interesses, as suas restrições ainda possuem certas carências em localizar interesses que envolvam restrições globais, que não são utilizadas por um módulo somente, ou considerando ainda módulos que possuam dependências de si (Almeida, 2005). A programação orientada a aspecto é uma nova evolução dentro da linha denominada tecnologia para separação de interesses. Sendo que, esta permite que projeto e o código possam ser estruturados de modo que os desenvolvedores possam pensar sobre o sistema. As funcionalidades espalhadas em um *software* são denominadas *crosscutting concerns* (Chitchyan, 2005).

Deve-se salientar também que o termo programação orientada a aspectos foi criado pelo grupo de pesquisa da Xerox PARC, o qual foi coordenado por Kiczales, sendo que as

características possíveis que são moduladas nos paradigmas de desenvolvimento estruturado ou orientado a objeto são denominadas componentes (Kiczles, 1997). Além disso, consideram-se aspectos, as unidades modulares que agrupam características em comum de componentes do sistema, de modo que estes podem ser utilizados em qualquer ponto não alterando a estrutura das classes do sistema (Kiczles, 1997). Assim, os aspectos diferentes dos componentes representam outra unidade de decomposição do sistema, podendo-se implementar os componentes em uma linguagem procedural ou orientada a objetos e os aspectos devem ser implementados em uma linguagem orientada a aspectos. A programação orientada a aspectos fica centrada na idéia de que os sistemas podem ser melhores desenvolvidos isoladamente pela especificação dos interesses. Com isso o projeto torna-se mais modular, facilitando a localização dos interesses e a relação destes com o resto do sistema.

No entanto, através da orientação a objetos torna-se complexa a atividade de implementar funcionalidades de forma otimizada, pois elas estão inseridas nos requisitos básicos de um *software,* não sendo possível, assim, separá-las em um mesmo componente ou classe. Porém, um dos maiores benefícios da programação orientada a objetos é o grau elevado de reuso. Considera-se que o reuso significa que um mesmo fragmento de código pode ser reaproveitado diversas vezes, consequentemente, existira menos código para dar manutenção, fazendo com que a produtividade aumente e o melhore o suporte as mudanças de requisitos. Embora a grande vantagem da orientação a aspectos seja exatamente a modulação das funcionalidades, pois ela auxilia os analistas a prover uma abstração da representação dos requisitos transversais, e, conseqüentemente, oferece mecanismos para descobrir aspectos em documentos de requisitos (Sampaio 2005). Além disso, muitas das tomadas de decisões estão fortemente relatadas e interdependentes, sendo praticamente impossível conduzir todas as saídas que se deve levar em conta ao mesmo tempo. Assim, visando controlar a complexidade de um projeto, há a necessidade de separar os interesses (Almeida, 2005).

Segundo Souza (Souza 2004 *apud* Ghezz 1991), a aplicação desse princípio traz alguns benefícios, os quais são:

- Diminuir a complexidade do desenvolvimento de *software*, concentrando-se em diferentes características separadamente. Sendo que a programação orientada a aspectos dirige-se ao mínimo de acoplamento de cada interesse. Este acoplamento produz um *software* com menos código duplicado.

- Limitar a quantidade sobre diferentes características, de maneira relativamente independente;

- Facilitar a inserção e a remoção de preocupações na aplicação. Considerando que os módulos e preocupações não precisam ter conhecimento dos interesses, torna-se fácil a inserção criando novos aspectos.

- Dividir esforços e separar responsabilidades entre membros da equipe de desenvolvimento;

- Melhorar a modulagem de artefatos de *software.*

A Figura 3 ilustra a funcionalidade da programação orientada a aspecto, que representa a política de segurança do *software* separada em seis classes, sendo que, nesse caso típico, existe a necessidade de modular o *software* em apenas um fragmento de código.

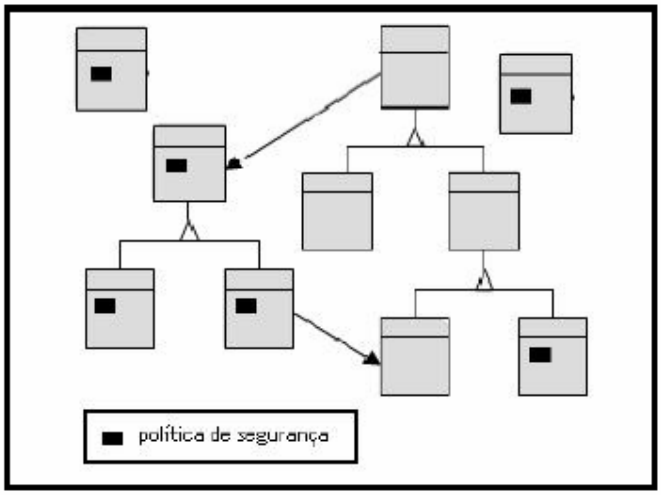

Figura 3 : Funcionalidade dispersa

Além do mais, as linguagens orientadas a aspectos utilizam cinco elementos que modulam as características transversais, sendo eles (Elrad, 2001):

- Um modelo de pontos de junção que descreve possíveis pontos, no qual o comportamento do aspecto pode ser adicionado;

- Um meio de identificar esses pontos de junção;

- Um meio de especificar o comportamento adicional desses pontos;
- Unidades que encapsulam a especificação dos pontos de junção;
- Um mecanismo para combinar o comportamento transversal do aspecto com o comportamento do componente que ele afeta.

Deve-se observar ainda, que as linguagens de programação orientada a aspectos utilizam um processador de aspectos, denominado *weaver*, e são implementados através de uma unidade denominada *aspect.* Sendo que o código relacionado às características transversais utiliza os seguintes construtores:

- *Pointcut*: define-se onde o aspecto irá atuar, mas não quais serão suas atribuições de atuação do mesmo;
- *Advice*: representa os "métodos" que dependem da chamada de um *pointcut* e é executado em tempo de execução (*runtime*), ou seja, é invocados quando um *pointcut* que está relacionado a ele for atingido. Tais métodos podem ser utilizados em vários modos, tais como: *after*, *after returning*, *after throwing*, *before*, e *around.* Ou seja, essas regras indicam que o comportamento do aspecto deve ser executado, respectivamente, depois, antes ou ao redor;
- *Inter*-type d*eclarations*: são adições às classes existentes, ou seja, significa adicionar declarações extras a uma classe ou a um objeto, ajustando-o às necessidades do programador em relação a um aspecto.

A Figura 4 apresenta um pequeno fragmento de código implementado através da linguagem AspectJ, cujo código apresenta a implementação do aspecto denominado *Log*.

```
1   public aspect Log {
2
3     pointcut logMethod() : call(* *(..));
4
5     before() : logMethod() {
6               System.out.println("Antes da execução");
7     }
8
9     after() : logMethod()  {
10             System.out.println("Após execução");
11    }
12  }
```

Figura 4 : Fragmento de código do aspecto Log

Observa-se, que na linha 3 da Figura 4, é definido o *pointcut* denominado *logMethod*, que inclui todas as chamadas de método. Já na linha 5 e 9 apresentam os *advice* que, por sua vez, apresenta a execução, antes e após a chamada do método.

Também é importante citar que as dificuldades mais comuns encontradas na orientação a aspectos são a falta de processos e de técnicas que possam auxiliar os usuários a aplicar os fundamentos deste paradigma. Portanto, essa atividade depende da experiência e da intuição do desenvolvedor. No entanto, existem algumas propostas para a elicitação de aspectos através dos requisitos como PREview, *Viewpoints and Incosistency Management* (VIM), *Non-Functional Requirements Framework* (NFRF), *KAOS Method, I**, *Problem Frames, Use Case Method, ARCADE*, *COSMOS* e *Theme/DOC* (Chitchyan, 2005).

Então, considerando as abordagens citadas acima, nota-se que seu foco é elicitar aspectos em nível de análise, denominada Engenharia de Requisitos Orientada a Aspectos, cujas principais atribuições são (Chitchyan, 2005):

- Identificar e modelar aspectos em nível de requisitos;
- Integrar e compor os aspectos com outros mecanismos de modelagem de requisitos;
- Rastrear aspectos identificados no nível de requisitos e nas atividades seguintes de desenvolvimentos.

Porém o paradigma de desenvolvimento a aspectos ainda está em fase de desenvolvimento, visto que ainda não existe um consenso a respeito dos conceitos e práticas que ele inclui, embora as linguagens que suportam as implementações de aspectos já estejam bem definidas. Por outro lado, as metodologias, a serem utilizadas nas fases de análise e projeto ainda são objetos de pesquisa e encontram-se em fase inicial, uma vez que existem muitas propostas, porém pouca coerência e continuidade nas mesmas (Souza, 2004).

Portanto, neste contexto, o foco do trabalho é a utilização desses artefatos de representação dos aspectos na fase de análise, a fim de incluir os mesmos como elementos para mensurar o custo de um projeto, sendo que, na próxima seção são apresentados dois trabalhos que utilizam os artefatos da UML para mensurar o custo de um projeto.

3. TRABALHOS RELACIONADOS

Este capítulo tem por objetivo apresentar dois trabalhos relacionados à estimativa de custos nos quais foram utilizados artefatos da UML, e que se assemelham com à metodologia AspectCost em alguns fatores, sendo que é apresentado um comparativo entre os três projetos. A escolha dos trabalhos aqui apresentados e comparados com a metodologia AspectCost estão diretamente relacionados a alterações das metodologias já existentes, como UCP e a utilização de ferramentas de apoio na metodologia. A comparação entre os trabalhos beneficia uma análise dos benefícios e melhorias que a metodologia AspectCost pode usufruir para consolidar o seu processo de estimativa.

3.1 *EFFORT ESTIMATION OF USE CASES FOR INCREMENTAL LARGE-SCALE SOFTWARE DEVELOPMENT*

Para adaptar o modelo de estimativa por casos de uso para grandes projetos, Mohagheghi *et. al.* propuseram uma extensão do original, sendo que o fator motivador para o desenvolvimento do trabalho foi a grande necessidade de métodos ágeis e incrementais aplicáveis em projetos de desenvolvimento de *software* (Mohagheghi, 2005). Assim, nesses modelos, requisitos são descobertos em sucessivos *releases,* e a mudança dos requisitos é aceita como fator chave para o desenvolvimento. Portanto, visando a validar o modelo apresentam-se alguns estudos empíricos nos quais utiliza-se a abordagem proposta, sendo que o modelo modificado seguiu os passos descritos a seguir:

Passo 1 – Atores dos casos de uso (AUW):

Os atores podem ser humanos, assim como outros sistemas ou protocolo. A classificação dos mesmos tem um pequeno impacto no resultado final da estimativa, por isso assume-se um valor médio para os atores, sendo que o cálculo da modificação dos atores é determinado a partir do *Modified Unadjusted Actor Weights* (MUAW).

- 1.1 UAW = #Actors * 2
- 1.2 MUAW = #New or Modified actors * 2

Passo 2 – Cálculo do UUCW e MUUCW:

Para trabalhar de uma forma mais precisa, os casos de uso são separados em pequenos fragmentos e os cálculos seguem as seguintes regras básicas, as quais são:

- R1) Cada transação no fluxo principal contém uma ou mais transações alternativas, assim, conta-se cada uma delas como se fosse um caso de uso;
- R2) Conta-se cada fluxo alternativo como se fosse um caso de uso;
- R3) Para fluxos alternativos, parâmetros e eventos é atribuído peso 2. Portanto, a soma dos pesos não deve atingir mais do que 15, sendo considerado como caso de uso complexo;
- R4) Deve-se incluir os casos de uso com os estereótipos <<include>> e <<extended>>;
- R5) Deve-se classificar os casos de uso seguindo as mesmas regras do modelo original.

Desta forma, a Figura 5 apresenta um exemplo prático da abordagem descrita.

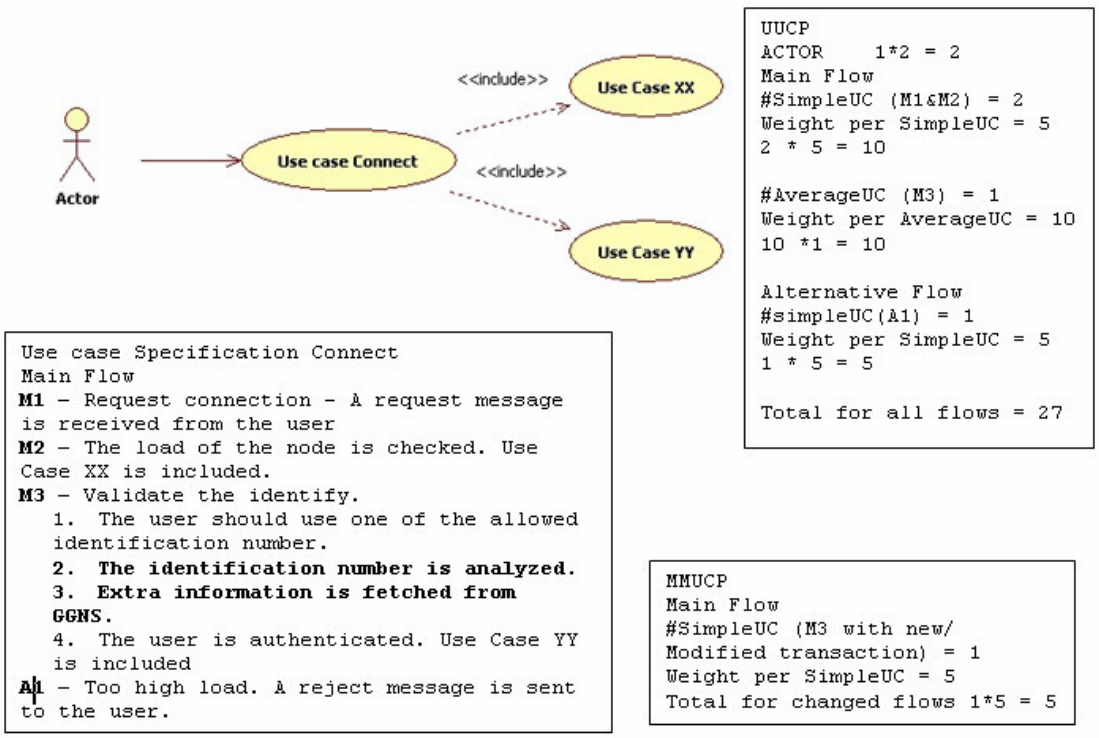

Figura 5 : Exemplo de *Use Case*

Observa-se que no caso de uso *Connect* representado pela Figura 5, existem três fluxos no fluxo principal (M1, M2 e M3) e um no fluxo alternativo (A1), sendo que M1 representa um fluxo que descreve a verificação de uma mensagem de usuário; M2 se refere ao caso de uso com o estereótipo <<include>>; e M3 contém quatro transações sendo que uma delas inclui um outro caso de uso. Já considerando o passo 2, regra R5, cada uma das transações é classificada como se fosse um caso de uso.

Para calcular as transações alteradas ou parâmetros, deve-se contá-las. Assim, observando a Figura 5, nota-se que as transações M2 e M3, cujos fluxos estão em negrito (2 e 3), indicam que eles foram modificados ou adicionados, e, então, são classificados como simples. Já o cálculo do UUCP e MUUCP segue respectivamente as fórmulas (12) e (13):

$$UUCW = \sum (\#\text{Use Cases} * \text{Peso}) + \sum (\#\text{Use Cases for exceptional flows}) \quad (12)$$

$$MUUCW = \sum (\#\text{New/Modified Use Cases} * \text{Peso}) + \sum (\text{Points for new/modified exceptional flows}) \quad (13)$$

A partir da regra R5 obtém-se o valor do MUUCW, sendo 5/27 = 0.19, em que 27 é o total de fluxos e 5, o total de fluxos alterados multiplicado por 0.19 (0.19*5 = 0.95).

Passo 3 – Cálculo do UUCP:

O resultado do UUCP é obtido a partir da fórmula do modelo original (14):

$$UUCP = UAW + UUCW \quad (14)$$

Sendo que neste passo deve ser considerado o cálculo do MUUCP que é obtido pela soma do MUAW e MUUCW (15):

$$MUUCP = MUAW + MUUCW \quad (15)$$

Passo 4 - Fatores técnicos de ambiente:

Dentro da abordagem assume-se o valor 1 para ambos.

$$TCF = EF = 1$$

Passo 5 – Cálculo do *Adjust Use Case Point* (UCP) e *Adjusted Modified UCP* (MUCP) (15):

Portanto, como os valores dos fatores técnicos e de ambiente são 1, os valores do UCP e MUCP são os mesmos valores de UUCP e MUUCP.

$$UCP = UUCP$$

$$MUCP = MUUCP$$

Passo 6 - Estimativa do esforço:

Na abordagem existem dois mecanismos que exigem esforços, uma é a estimativa de esforço para os casos de uso novos e modificados, e a outra estimativa para as segundas alterações do *software*, *E_primary* e *E_secondary*, respectivamente.

Sendo que o *E_primary* é estimado através no número de pessoas/horas por casos de uso multiplicado pelo MUCP (16):

$$E\_primary = MUCP * PHperUCP \quad (16)$$

Por sua vez, a estimativa das alterações é obtida através da equação (17):

$$E\_secondary = (UCP - MUCP) * EMF * PHperUCP \quad (17)$$

Já a estimativa total é a soma desses dois elementos (18):

$$E = E\_primary + E\_secondary \quad (18)$$

O EMF (*Equivalent Modification Factor)* é composto pelos fatores que influenciam na modificação de um *software,* os quais foram herdados do modelo de estimativa de projeto COCOMO, com seus respectivos valores, que são:

- Avaliação: 2% (assume-se baixa procura, teste e avaliação de esforço);
- Entendimento do *software:* 30%;
- Modificação de código: 55%;
- Modificação de projeto: 30%;
- Integração de esforços: 65%;
- Não familiaridade com o *software*: 20%.

3.2 *FAST && SERIOUS: A UML BASED METRIC FOR EFFORT ESTIMATION*

O modelo foi proposto por Carbone, que estima o esforço de um *software* através dos diagramas da UML, ou seja, esse modelo estima o *software* a partir do número de linhas de código. Além disso, a metodologia define como artefato básico e mais importante o diagrama de classe da UML, sendo que a ferramenta importa o diagrama desenvolvido pelo *Rational Rose 2002* e realiza uma série de passos até o resultado final. A Figura 6 apresenta um fluxo macro da metodologia (Carbone, 2002).

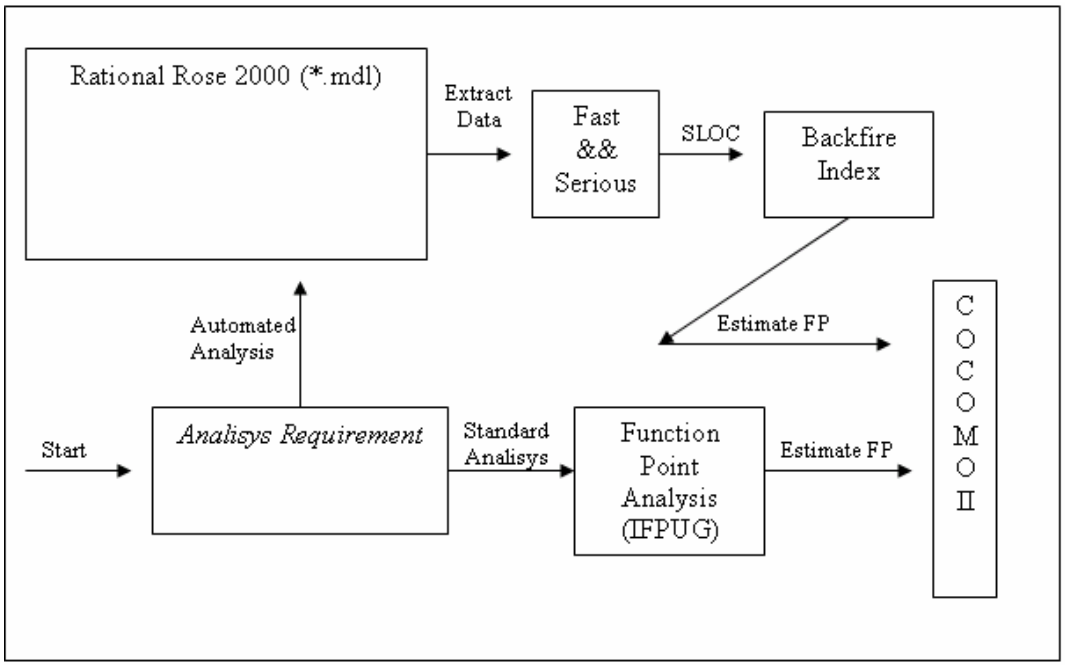

Figura 6 : Fluxo do modelo (Carbone, 2002).

Observa-se na Figura 6 que as linhas de código são transformadas em tradicionais Pontos por Função utilizando *Backfire Index,* o que permite estimar os pontos por função e inserir no modelo COCOMO II. Além disso, nota-se que os pontos por função são estimados como nos antigos modelos, ou seja, através dos requisitos e não dos diagramas UML.

É importante observar também que o método inicia com a análise nos diagramas de classe e que o processo de estimativa consiste em seis passos que serão descritos a seguir e dos quais três são opcionais.

Passo 1 – Análise do diagrama de classe e extração das seguintes métricas:

- MPC – Número de métodos por classe;
- NAC – Número de associação por classe;
- PMS – Porcentagem de métodos com assinatura;
- DIT – Profundidade da árvore de inerência.

Passo 2 – Computando a complexidade do diagrama de classe:

Calcula-se o valor da complexidade de cada uma. Sendo que primeiro devem ser classificados os atributos em categorias conforme a Tabela 11:

Tabela 11 : Classificação de atributos

Complexidade	Definição	Peso
Leve	Atributo simples, como inteiro, string, etc.	1
Pesado	Atributos complexos desenvolvidos e testados, os quais são definidos em pacotes.	3
Importado	Atributos complexos não desenvolvidos e não testados.	5

Nota-se que todo o conjunto dos atributos é organizado em tuplas, <atributo,tipo>, e que o cálculo do ponto de estado é realizado através da fórmula (19):

$$SP(c) = \sum_{i=1}^{numAttr(c)} Peso(ListAttr_i) \quad (19)$$

Sendo *numAttr(c)* o número de atributos oriundos de uma classe *c*.

Assim, a próxima etapa é classificar os métodos em “triviais” ou “substancial”, de acordo com as definições apresentadas na Tabela 12:

Tabela 12 : Classificação de Métodos

Complexidade	Definição
Trivial (T)	Métodos que têm mais de 80% dos atributos classificados como leve e não possuem nenhum atributo do tipo importado.
Substancial (S)	Não contém métodos triviais.

Então, para cada método *m* calcula-se a complexidade do mesmo (CM) (20):

$$CM(m) = (T \quad S) * \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \end{pmatrix} * \begin{pmatrix} numLA \\ numHA \\ numIA \end{pmatrix} \quad (20)$$

Sendo *numLA, numHA, numIA,* os número de atributos leves, pesados e importados em um método *m*, respectivamente. Por sua vez, *T* e *S* são os padrões para Trivial (T=1, S=0) e Substancial (T=0, S=1). Além disso, utiliza-se o CM pra estimar o ponto de comportamento (*Behavioural Point – BP)* de uma classe *c***,** conforme a equação (21):

$$BP(c) = [1 + numAss(c)] * \sum_{i=1}^{numMet(c)} CM(m_i) \quad (21)$$

Onde, *numAss(c)* é o número de classes associadas (NAC) e *numMet(c)* é o número de métodos da classe c.

Por fim, calcula-se o número de pontos de classe (*Class Point – CP*) de acordo com a expressão (22):

$$CP(c) = 2 * SP(c) + 3 * BP(c) \quad (22)$$

Passo 3 – Complexidade dos diagramas de caso de uso:

Tratando do diagrama de caso de uso, atribui-se pesos para os atores e para os casos de uso.

Desta forma, para cada caso de uso *uc* é associado o número de fluxos (índice *numScen*), a fim de ser obtida a complexidade do mesmo (COUC), como mostra a equação (23):

$$COUC(uc) = 1 + \sum_{i=1}^{numScen(uc)} numMess_i \quad (23)$$

Sendo, *numMess* o número de mensagens entre todos os objetos de um fluxo. Já para os atores, a complexidade é calculada através da fórmula (24):

$$COA(a) = numAss(a) * dit(a) \quad (24)$$

Sendo, *numAss(a)* o número de associações entre um ator *a* e o caso de uso, e *dit(a),* a profundidade da árvore de inerência. Assim, a abordagem assume que um ator *a* é associado a uma classe *c,* se o caso de uso está associado com a referida classe *c.* Por conseguinte, para a classe *c* calcula-se a complexidade do casos de uso associado à classe (CUCA) e a complexidade do ator associado (CAA), através das fórmulas (25) e (26), respectivamente:

$$CUCA(c) = \sum_{i=1}^{numUC_{(c)}} COUC(uc_i) \quad (25)$$

$$CAA(c) = \sum_{i=1}^{numAct(c)} COA(a_i) \quad (26)$$

Sendo, *numUC(c)* o número de casos de uso associados a uma classe *c* e numAct(c) o número de atores associados com à mesma classe *c.*

Passo 4 – Explorando o diagrama de interação:

Para cada diagrama de seqüência sed_j onde j=1..*nsed,* sendo *nsed* é o número de diagramas de seqüência. Em seguida, analisa-se cada instância de uma classe *c* em cada sed_j calculando *numMess(c,* sed_j *)*, sendo o número de mensagens enviadas ou recebidas. Por fim, calcula-se o *totMess(c)* através da soma dos *numMess(c,* sed_j *)*, em que *rsed* é a razão entre o número de diagramas de seqüência em uma classe *c.* Conseqüentemente, o valor obtido é utilizado para localizar a porcentagem de incremento ou decremento do CP, conforme pode-se notar na Tabela 13.

Tabela 13 : Porcentagem do CP

toMess(c) / *rsed(c)*	0-3	4-6	7-9	10-13	>13
0-19%	-10%	-5%	5%	7%	12%
20-49%	-5%	2%	5%	10%	13%
50-100%	4%	5%	7%	10%	15%

Passo 5 – Diagrama de estados:

Para um determinado diagrama de estados *std* associado a uma classe *c* ou a um método da classe *c*, extrai-se o número de estados *numStat(std)* e o número de ações *numAction(std).* Assim, após obter os valores calculados, verifica-se na tabela a relação, ou seja, se eles incrementam ou diminuem o valor de CP, como pode ser observador na Tabela 14:

Tabela 14 : Porcentagem do CP

totStat(std) / *totAction(std)*	0-3	4-6	7-9	10-13	>13
0-5	-8%	-3%	3%	7%	12%
6-10	-3%	2%	4%	10%	13%
11-20	3%	4%	7%	12%	15%

Passo 6 – Estimando o tamanho do sistema:

Para a estimativa do tamanho das linhas de código de um *software* em Java (Carbone, 2002), utiliza-se respectivamente as fórmulas (27) e (28):

$$Size(c) = 4 + 10 * CP(c)^{0.7} \quad (27)$$

$$Size(System) = \sum_{c_{i \in CD}} Size(c_i) \quad (28)$$

Portanto, a estimativa é totalmente voltada para a abordagem que utiliza a UML como artefato de representação, embora ainda existam algumas limitações com relação ao resultado gerado e à automação do processo.

3.3 CONSIDERAÇÕES SOBRE OS TRABALHOS RELACIONADOS

O AspectCost é considerada uma metodologia de estimativa voltada para o paradigma de desenvolvimento orientado a aspecto, e que unifica qualidades existentes, como demonstrados nos trabalhos apresentados. Sendo assim, a Tabela 15 apresenta um comparativo entre as três abordagens, seguida da justificativa de desenvolvimento da pesquisa.

Tabela 15 : Comparação entre os trabalhos relacionados

	Effort Estimation of Use Cases for Incremental Large-Scale Software Development	*Fast && Serious*	**AspectCost**
Métodos ágeis (RUP, XP, etc.)	X		X
Consideração de aspectos			X
Metodologia com ferramenta de apoio.		X	X
Gerenciamento de requisitos			X
Adaptação ao COCOMO II		X	X
Rastreabilidade dos valores ao longo das atividades da metodologia			X
Extensão do método Use Case Point	X		X

Após feita a comparação entre os trabalhos relacionados, pode-se concluir que assim como a metodologia *Effort Estimation of Use Cases for Incremental Large-Scale Software Development,* o AspectCost estende a metodologia de estimativa *Use Case Point*. Sendo que, para realizar essa estimativa, os trabalhos mantêm a estrutura da metodologia, porém customizam-na de acordo com a necessidade que interessa em estimar esforços.

É possível notar que a adaptação a métodos ágeis identifica as mudanças em cada interação, mas que, porém, o AspectCost utiliza-se deste mecanismo para a rastreabilidade dos valores ao longo das fases de desenvolvimento do método. Além disso, o AspectCost considera os aspectos como fatores de desenvolvimento, que interferem no valor do custo total do projeto.

O trabalho *Fast&&Serious* é a utilização da UML para a modelagem dos sistemas e a automação de todo o processo. Este, não mensura os requisitos do *software* e não provê um rastreamento dos valores. A sua grande limitação é que a metodologia estima o esforço baseado em linhas de código, fator esse que é bastante variável nas diversas linguagens de desenvolvimento. Já a relação com o AspectCost é a adaptação do valor estimado através da metodologia ao COCOMO II.

4. METODOLOGIA ASPECTCOST

Este capítulo descreve uma metodologia para estimativa de projeto e controle de custos, denominada AspectCost, que apresenta uma forma de estimar o custo de um projeto e controlar valores a partir da variação dos requisitos, podendo-se considerar projetos desenvolvidos sobre o *framework* RUP em um contexto de desenvolvimento orientado a aspectos.

A base para o desenvolvimento da estimativa está direcionada para a metodologia *Use Case Points*, devido à utilização da linguagem UML, como forma de representação voltada ao paradigma de desenvolvimento orientado a objetos. De acordo com Karner, a adoção da estimativa baseada nos diagramas de caso de uso possibilita estimar o custo do projeto desde as fases iniciais de desenvolvimento, objetivando, assim, o planejamento facilitado e o controle durante todo o ciclo (Karner, 1993).

A metodologia AspectCost possui três características que são consideradas como principais contribuições da pesquisa aqui desenvolvida e apresentada:

- Estimativa do custo total do projeto: a principal contribuição é englobar os casos de uso aspectuais propostos por Souza para propiciar uma estimativa de projeto para os que utilizam a programação orientada a aspectos (Souza, 2004);

- Estimativa dos custos dos requisitos: baseada no processo *Analytical Hierarchy Process* (AHP), proposto por Saaty, a metodologia AspectCost utiliza esse processo para quantificar as características qualitativas, as quais são, nesse contexto, os requisitos. Sendo que o principal ganho na utilização desse processo é a possibilidade de associar custo a cada requisito que faz parte do projeto, o resultado final é o valor, em percentual, de cada requisito, se considerarmos que um projeto é composto por N requisitos (Saaty, 1991).

- Controle dos custos dos requisitos: uma vez que os requisitos não são isolados ao fazerem parte de um projeto, muitos deles se relacionam entre si, de acordo com as funcionalidades especificadas e exigidas pelos clientes. Visando a sanar esse problema, utiliza-se a tabela comparativa de relacionamento proposta por Rashid, e os

fatores de relacionamento entre os mesmos para controlar as alterações de valores ao longo dos requisitos e, conseqüentemente, do projeto (Rashid 2003).

A Figura 7 apresenta a metodologia sob uma visão macro, representada através do diagrama de atividades da UML, e cujas fases são descritas a seguir.

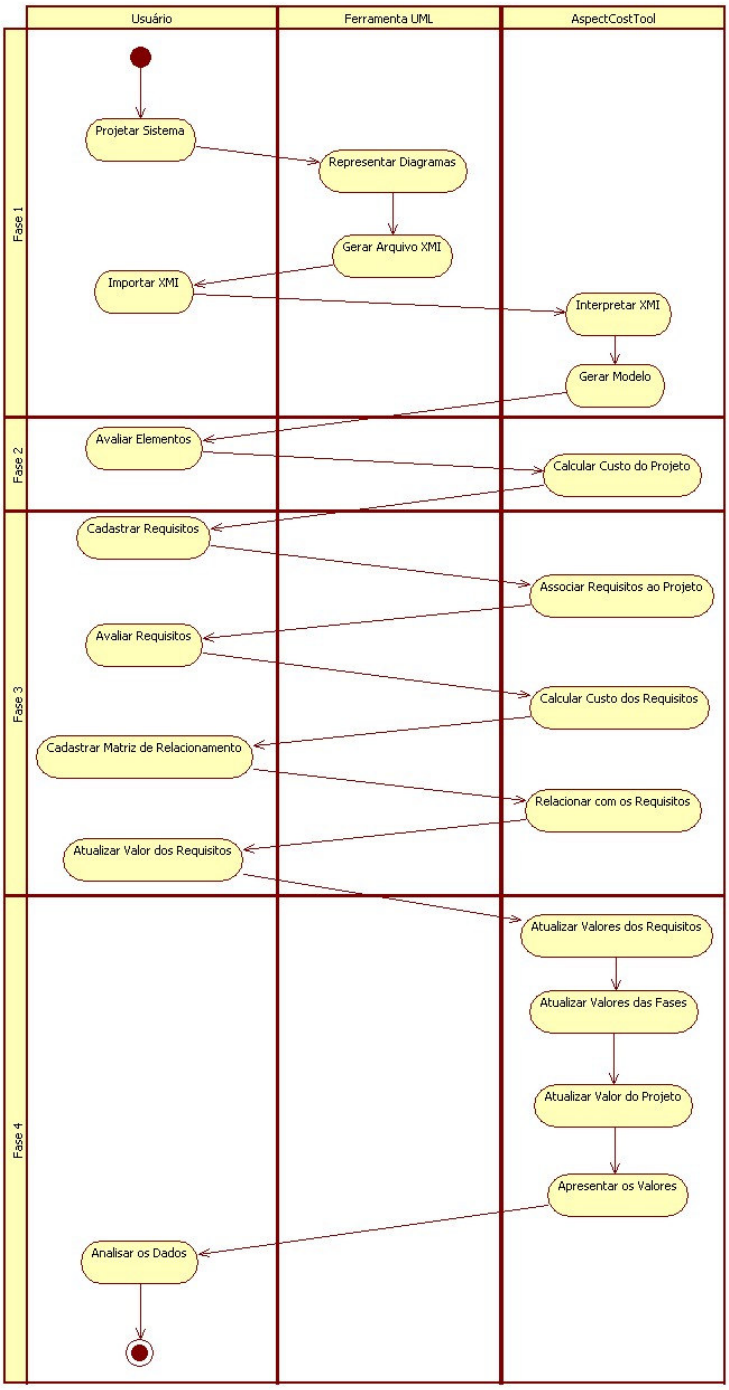

Figura 7 : Diagrama de atividades da metodologia

4.1 FASE 1: PROJETAR DIAGRAMAS DE CASOS DE USO

Essa fase da metodologia engloba as seguintes atividades: projetar sistema, representar diagramas, gerar arquivo XMI, importar XMI, interpretar XMI e gerar modelo.

A proposta de Souza em utilizar estereótipos, denominados aspectual, nos diagramas de casos de uso para modular um requisito não funcional do sistema prevê a antecipação na modelagem, de aspectos, para atividades de análise e projeto e, propicia então, a integração do paradigma com ferramentas e processos existentes (Souza, 2004).

A Figura 8 apresenta um caso de uso aspectual que corta um ou mais casos de uso, representados pelo estereótipo <<Aspectual>>.

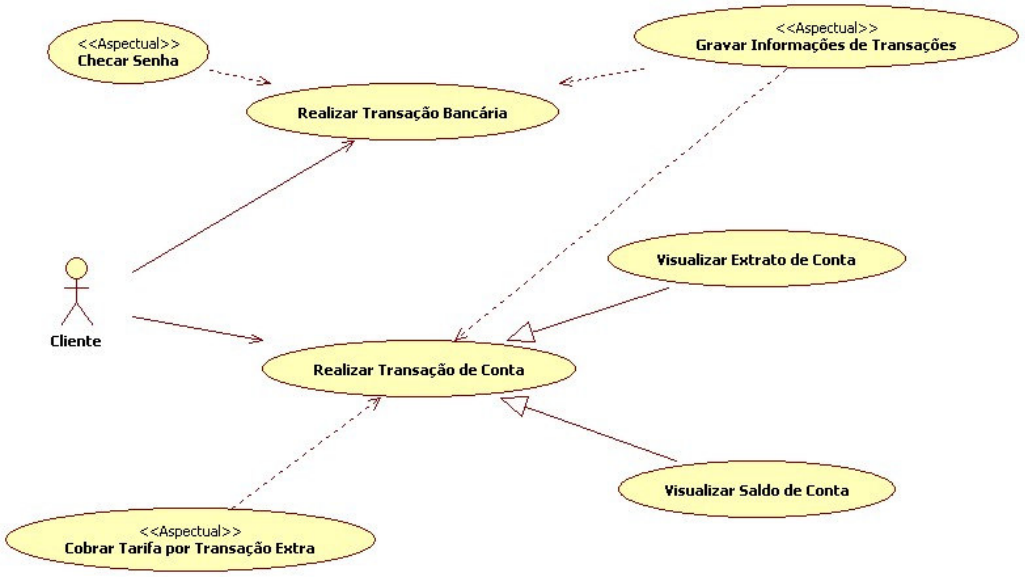

Figura 8 : Diagrama de caso de uso aspectual

Observando a Figura 8 verifica-se que o aspecto “Gravar informações de transição”, afeta os casos de uso “Realizar transição bancária” e “Realizar transação de conta”. Assim, a abordagem propõe a utilização de uma tabela para representar a composição na perspectiva do aspecto, ou seja, para indicar quais são as unidades afetadas por um determinado aspecto e como ele afetará cada uma delas. Com esse objetivo, a Tabela 16 apresenta as informações do aspecto “Gravar informações de transações”. Os valores apresentados na coluna Pontos de Junção são meramente ilustrativos, considerando que estas informações não são retiradas do diagrama de caso de uso, apenas na etapa de expansão.

Tabela 16 : Representação da composição na perspectiva do aspecto

Caso de uso aspectual: gravar informações de transição				
CASO DE USO AFETADO	CONDIÇÃO (OPCIONAL)	REGRA DE COMPOSIÇÃO	PONTOS DE JUNÇÃO	INFORMAÇÕES ADICIONAIS
UC realizar transação bancária	-	depois	7	-
Visualizar saldo de conta	-	depois	2	-
Visualizar extrato de conta	-	depois	4	-

Portanto, após essa atividade, o responsável pelas estimativas do projeto dispõe de todos os artefatos necessários para calcular o custo total do projeto.

4.2 FASE 2: CALCULAR O CUSTO TOTAL DO PROJETO

Essa fase da metodologia contempla as atividades de avaliar elementos e calcular custo total do projeto, as quais são apresentadas no diagrama de atividades da Figura 7. Além disso o cálculo total do projeto é realizado a partir da adaptação do modelo *Use Case Points*, a fim de considerar os casos de uso aspectuais, sendo que a adaptação é feita a partir da Tabela 2, na seção 2.2.1. Assim, a medida da funcionalidade do sistema é baseada no cálculo denominado *Unadjusted Use Case Point*, apresentado na seção 2.2, cuja fórmula é:

$$UUCP = UAW + UUCW \quad (29)$$

Sendo que UAW (*Unadjusted Actor Weight*) determina a complexidade dos atores que fazem parte do diagrama, e que UUCW (*Unadjusted Use Case Weight*) define a complexidade dos casos de uso. Por sua vez, a complexidade dos atores está diretamente relacionada ao tipo no *Use Case Point*, considerando que o usuário humano tem maior capacidade de prever situações indesejadas em uma aplicação e, conseqüentemente, é maior a

probabilidade de encontrar falhas. Além disso, a complexidade dos casos de uso relaciona-se diretamente com o número de passos do fluxo principal e dos alternativos; e, considerando o desenvolvimento orientado a aspecto e a utilização dos casos de uso aspectuais propostos por Souza, adiciona-se mais um elemento para a soma do UUCP, o qual é denominado *Unadjusted Use Case Aspect Weight* (UUCAW) e será parte da soma do valor da funcionalidade do sistema. Sendo assim, a fórmula (30) do UUCAW foi determinada seguindo os princípios dos outros dois elementos:

$$\text{UUCAW} = \sum \left(\#\, Casos de uso Aspectual * Peso \right) \quad (30)$$

Ainda é importante salientar que a atribuição do peso para cada um dos casos de uso aspectuais é determinada pelo somatório de todos os pontos de junção que o aspecto afeta, conforme apresenta-se na Tabela 17.

Tabela 17 : Pontos de junção

Complexidade	Definição	Peso
Simples	Menor do que 5. O caso de uso aspectual que detém até 5 pontos de junção, considerando a soma de todos que são afetados.	5
Médio	De 5 a 10. O caso de uso aspectual que detém de 5 a 10 pontos de junção, considerando a soma de todos que são afetados.	10
Complexo	Mais do que 10. O caso de uso aspectual que detém mais de 10 pontos de junção, considerando a soma de todos que são afetados.	15

Então, tomando como base a Tabela 16, nota-se que na composição do caso de uso aspectual “Gravar informações de transição”, o somatório dos pontos de junção resulta em 13, cuja classificação enquadra-se como complexo e o peso atribuído é 15 (Tabela 17).

Portanto, considerando o cálculo do UUCP e atribuindo o elemento UUCAW ao mesmo, a fórmula resultante é (31):

$$UUCP = UAW + UUCW + UUCAW \quad (31)$$

Os valores calculados serão monitorados e atualizados, caso haja alteração em qualquer um dos requisitos que fazem parte dessas fases, e, posteriormente será realizado o mapeando dos mesmos para determinar o custo total do projeto. Assim, na próxima seção, apresentam-se os fatores técnicos e de ambiente fornecidos pela metodologia *Use Case Point* que merecem atenção especial, considerando o paradigma de desenvolvimento orientado a aspecto.

4.2.1 Fatores técnicos e fatores de ambiente

O cálculo do *Use Case Point* é influenciado pelos fatores de ambiente e técnicos que também são considerados e adaptados para o contexto de programação orientado a aspecto. Dessa forma, considerando os fatores propostos por Karner, os que merecem mais atenção para serem atribuídos pesos considerando o desenvolvimento orientado a aspecto são:

- Código reutilizável: com a união de funcionalidades espalhadas pelo código a tendência desse fator é ser extremamente relevante;
- Facilidade de mudança: a modulação de funcionalidades facilita a manutenção, visto que a alteração fica limitada apenas àquele fragmento de código que foi implementado;
- Eficiência: a alteração de comportamento de um sistema em tempo de execução torna-o mais eficiente;

- Requer treinamento especial: isso porque se tratando de um novo paradigma, a equipe pode encontrar algumas dificuldades iniciais;

- Linguagem de programação difícil: uma vez que se trata de uma nova linguagem, a equipe pode encontrar certas dificuldades.

Portanto, esses são alguns exemplos de fatores de ambiente e técnicos que devem ser mais bem estudados para a atribuição do corrente peso. No entanto, não se limitando apenas a eles, o usuário pode optar pelos demais fatores que a metodologia UCP propõe, sendo que na seção seguinte apresenta-se a estimativa de custos dos requisitos e o relacionamento entre eles.

4.3 FASE 3: ESTIMAR E DEFINIR RELACIONAMENTO ENTRE OS REQUISITOS

A fase 2 mostrou que a estimativa do custo do projeto será realizada a partir da adaptação do modelo *Use Case Point.* Além disso, para determinar o valor de cada um dos requisitos de acordo com a metodologia descrita utiliza-se do processo denominado *Analytical Hierarchy Process* (AHP), uma vez que o conhecimento dos custos dos requisitos auxilia os gestores no desenvolvimento de um processo com maior eficiência e efetividade para gerenciar os projetos (Karlsoon, 1997). Esta fase da metodologia inclui as atividades de avaliar requisitos, calcular custo dos requisitos, cadastrar matriz de relacionamento, e relacionar com os requisitos.

Utilizando o processo AHP é possível obter o valor, em percentual, de cada um dos requisitos. Depois disso, o valor estimado do projeto será divido entre todos os requisitos, considerando o custo percentual de cada um deles, sendo que a Figura 9 apresenta graficamente a separação desses valores:

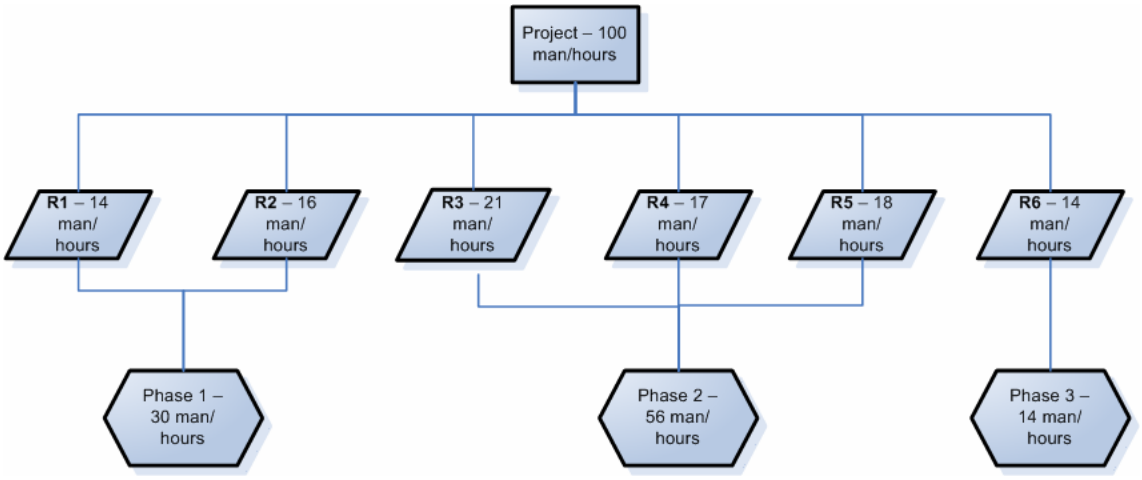

Figura 9 : Custo dos requisitos utilizando o processo AHP

Sendo assim, de acordo com Figura 9, observa-se os seguintes dados:

Custo total do projeto: 100 homens/horas.

Valor em percentual do requisito 1 (R1): 14% - Custo: 14 homens/horas

Valor em percentual do requisito 2 (R2): 16% - Custo: 16 homens/horas

Valor em percentual do requisito 3 (R3): 21% - Custo: 21 homens/horas

Valor em percentual do requisito 4 (R4): 17% - Custo: 17 homens/horas

Valor em percentual do requisito 5 (R5): 18% - Custo: 18 homens/horas

Valor em percentual do requisito 6 (R6): 14% - Custo: 14 homens/horas

Além disso, a Figura 9 permite observar que cada um dos requisitos do projeto pertence a uma fase, caso seja utilizado processos interativos de desenvolvimento. Assim, se somarmos os valores dos requisitos pertencentes a cada uma das fases, obtêm-se o valor de cada uma delas.

Já para controlar os custos dos requisitos optou-se pela tabela de relacionamento apresentada por Rashid (Rashid, 2003), que é aplicada conforme a Tabela 18:

Tabela 18 : Fatores de relacionamento

Relação	Valor
Fortemente relacionado	0,8 até 1,0
Relacionado	0,5 até 0,7
Medianamente relacionados	0,3 até 0,4
Pouco relacionado	0,1 até 0,2
Não relacionados	0,0

Os valores de comparação entre os requisitos são atribuídos através dos documentos de análise de projeto e caracterizados pelo grau de influência entre eles. Após a definição da matriz de relacionamento entre os diversos requisitos que fazem parte do projeto, é possível controlar a alteração ao longo de todas as etapas, conforme é apresentado na seção 4.4.

4.4 FASE 4: CONTROLE DOS CUSTOS

Considerando que os requisitos não estão isolados, há a necessidade de controlar a alteração de cada um deles ao longo do ciclo de vida do projeto, portanto, o principal objetivo dessa etapa da metodologia - que contempla as atividades de atualizar valor dos requisitos, atualizar valores dos requisitos, atualizar valores das fases, atualizar valor do projeto e apresentar os valores - é mapear o custo ao longo do projeto. Então, a Figura 10 apresenta o mapeamento dessas alterações, que se iniciam a partir da alteração do custo de algum dos requisitos.

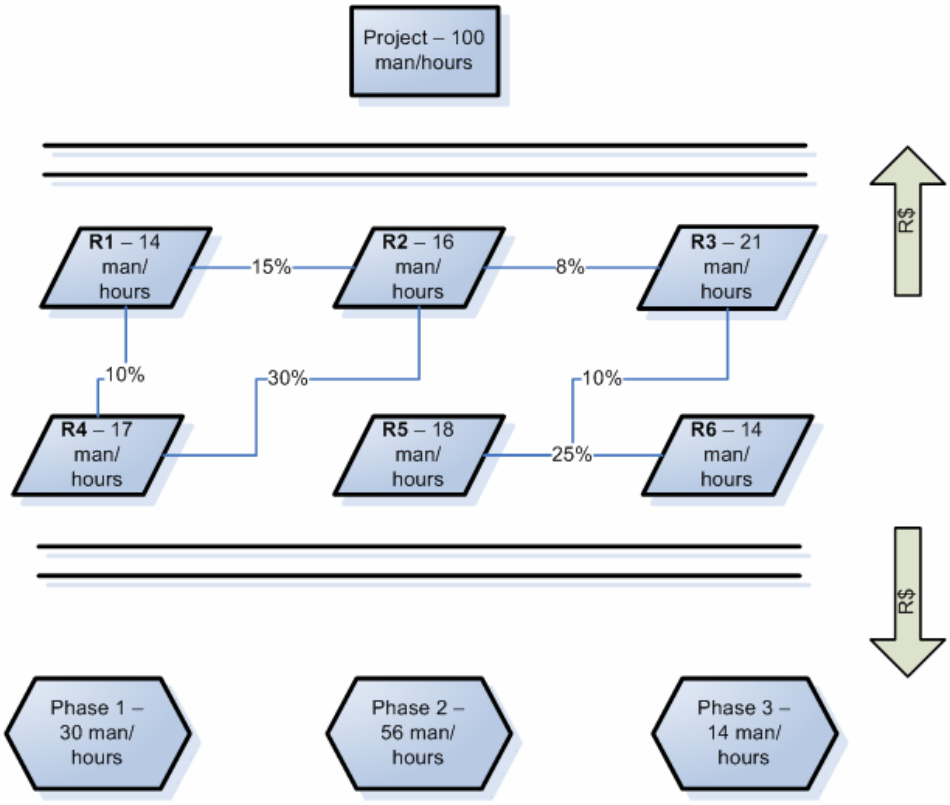

Figura 10 : Mapeamento dos Custos

É importante ainda salientar que a alteração no custo de um requisito pode impactar também no custo das fases dos requisitos alterados e, por fim, no total do projeto.

Assim, de acordo com a Tabela 19 observa-se os relacionamentos entre os requisitos.

Tabela 19 : Relacionamento entre os requisitos

	R1	R2	R3	R4	R5	R6
R1	X	15%	-	10%	-	-
R2		X	-	30%	-	-
R3	-	-	X	-	10%	25%
R4	-	-	-	X	-	-
R5	-	-	-	8%	X	-
R6	-	-	25%	-	-	X

O controle dos custos é efetuado através dessa matriz de relacionamento (Tabela 19), se o valor do requisito 1 aumentar de 14 homens/hora para 16 deve-se somar 10% do custo ao valor do requisito 2 e 15% do requisito 4. Exemplificando:

- Requisito 1: 14 homens/horas

- Novo valor do requisito: 1: 16 homens/horas

- Diferença: 2 homens/horas

- Novo valor do **requisito 2**: 16 + (2 * 0,1) = 16,2 homens/horas

- Novo valor do **requisito 4:** 17 + (2 * 0,15) = 17,3 homens/horas

Portanto, tem-se os seguintes valores resultantes para cada um dos requisitos:

- **Requisito 1:** 16 homens/horas

- **Requisito 2:** 16,2 homens/horas

- **Requisito 4:** 17,3 homens/horas

Sendo que, somando esses valores, obter-se-á o valor das fases das quais os requisitos fazem parte:

- Fase 1: 32,2 homens/horas

- Fase 2: 56,3 homens/horas

Finalmente, atribuindo-se ao custo total do projeto, o valor final será conforme a seguir:

Total do projeto: 32,2 + 56,3 + 14 = 102,5 homens/horas

Pelo resultado, observa-se, então, um acréscimo de 2,5 homens/horas no projeto usado como exemplo, que foi iniciado com 100 homens/horas.

Portanto, a alteração de valor de qualquer requisito será rastreada ao longo de todos os que se relacionam a ele. Assim, a solução utilizada na ferramenta que apóia toda a metodologia, apresentada no capítulo 5, são os grafos, sendo que, visando a automatizar a

metodologia, apresenta-se a ferramenta desenvolvida englobando todas as etapas apresentadas nesta seção.

5. ASPECTCOSTTOOL

Este capítulo apresenta a AspectCostTool, que se constitui no apoio da metodologia AspectCost, sendo que os procedimentos apresentados no capítulo 4 estão disponíveis nessa ferramenta, propiciando, assim, garantia nos procedimentos realizados e agilidade nas atividades. Além disso, a ferramenta foi desenvolvida em Java e segue o padrão de projeto MVC, que se constituiu no modelo 3 camadas, conforme pode-se observar na Figura 11.

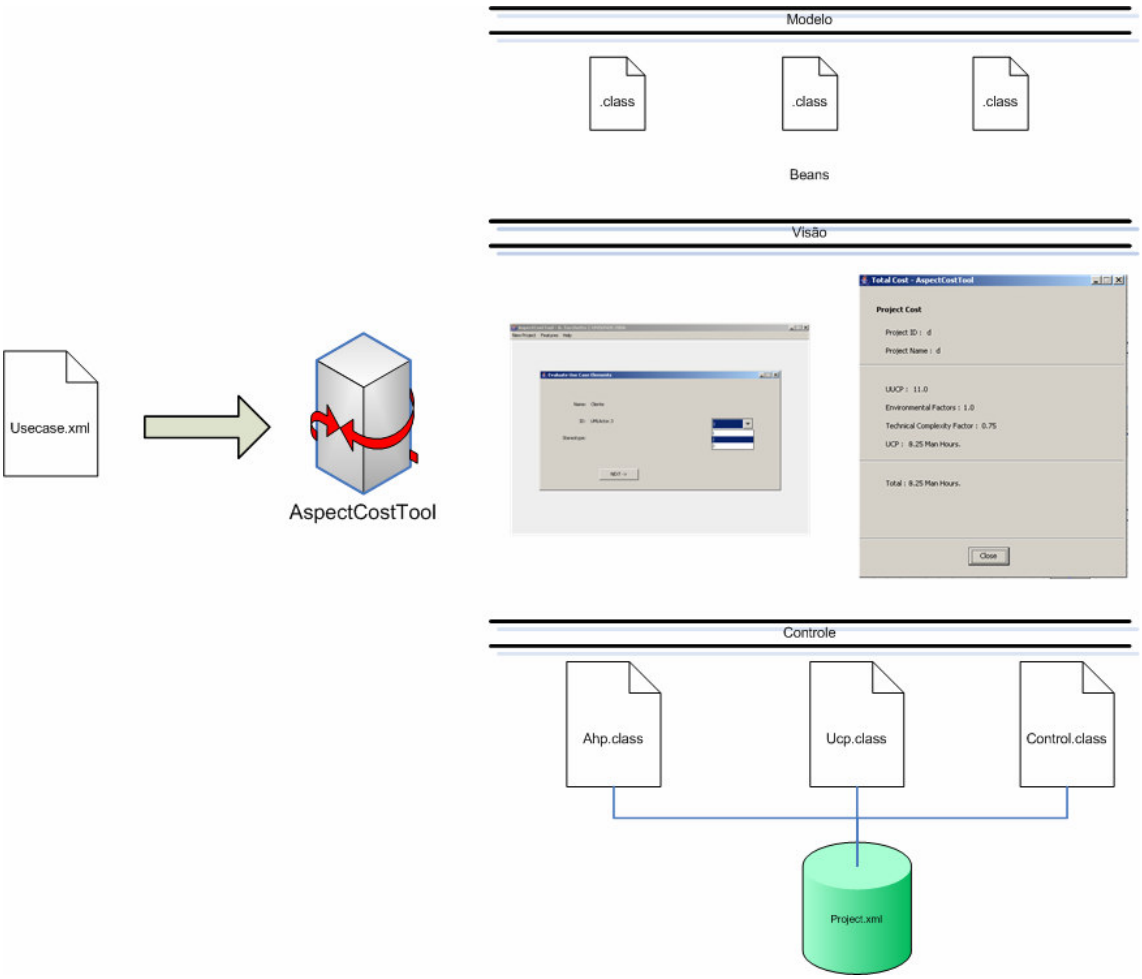

Figura 11 : Arquitetura da ferramenta AspectCostTool

Assim, pode-se observar que, na camada de modelo, os objetos da aplicação são representados pelo padrão de projeto JavaBeans, que é o termo utilizado para um modelo de componentes usados para criar aplicações com interface gráfica com o usuário, e o qual, segundo Horstmann é uma entidade com três capacidades (Horstmann, 2007):

- Habilidade para executar métodos;
- Habilidade para expor propriedades;
- Habilidade para gerar eventos;

Por sua vez, a camada de visão foi desenvolvida usando o pacote SWT da linguagem Java, mais especificamente o Swing, já a camada de controle foi implementada armazenando os dados em arquivos XML, sendo que o principal objetivo para o desenvolvimento da camada de persistência nesse tipo de arquivos é prever a reutilização dos dados gerados pela ferramenta a qualquer sistema que porventura deseja utilizá-los. Portanto, visando ao desenvolvimento da aplicação apresenta-se o diagrama de caso de uso (Figura 12) e a expansão dos mesmos, facilitando assim a documentação, e conseqüentemente, a implementação, conforme afirma Larman (Larman, 2007).

5.1 DIAGRAMAS DE CASOS DE USO

Os diagramas de caso de uso têm por objetivo apresentar o comportamento pretendido por parte do sistema desenvolvido, sem a necessidade de especificar o comportamento da implementação (Booch, 2000), diagrama esse que é representado pela Figura 12.

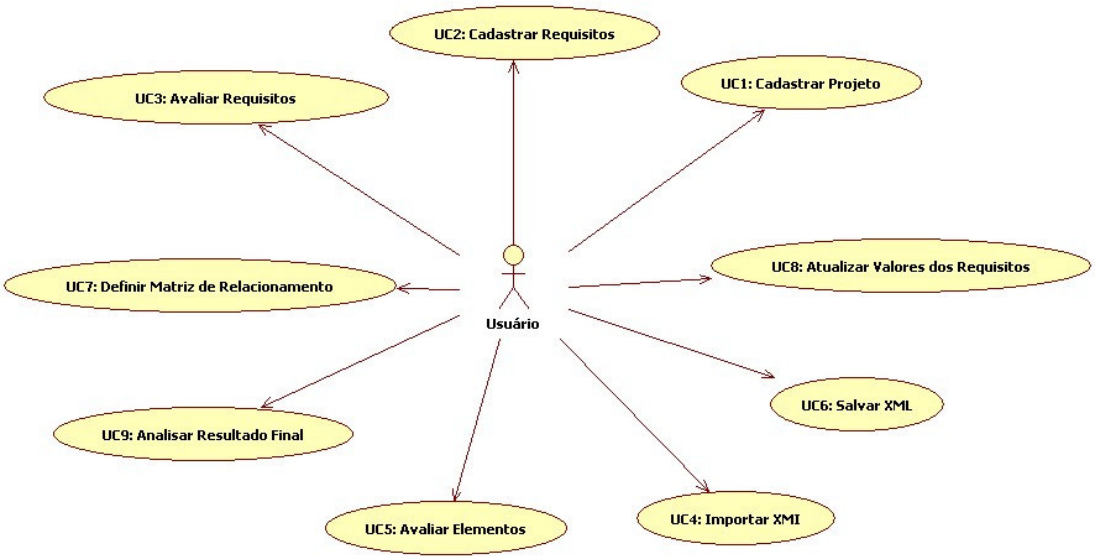

Figura 12 : Diagrama de casos de uso

Para melhor compreensão do sistema desenvolvido, a descrição de cada um dos casos de uso do modelo são apresentadas a seguir:

Primeiramente, a Tabela 20 apresenta o fluxo básico e o alternativo do caso de uso Cadastrar projeto, cujo objetivo é permitir ao usuário do sistema cadastrar o projeto para o qual deseja estimar os custos, assim como o dos requisitos que o integram.

Tabela 20 : UC1 – Cadastrar projeto

Fluxo básico:

1. Usuário seleciona o menu *NewProject;*
2. Usuário seleciona o menu *Start;*
3. Sistema apresenta a tela de cadastro de projeto;
4. Usuário preenche o campo *ID* do projeto;
5. Usuário preenche o campo *Name* do projeto;
6. Usuário pressiona o botão *OK*;
7. Sistema grava o projeto e emite uma mensagem de dados inseridos com sucesso;

Fluxos alternativos:

7.1 Se o campo *ID* ou *Name* estiver em branco, o sistema emite uma mensagem exigindo que os campos sejam preenchidos, pois devem ser obrigatórios.

Pré-condições: -

Pós-condições: projeto cadastrado.

Depois disso, para permitir ao usuário do sistema o cadastro dos requisitos integrantes do projeto, descreve-se o fluxo básico e os alternativos do caso de uso Cadastrar requisitos, conforme pode-se notar na Tabela 21.

Tabela 21 : UC2 – Cadastrar requisitos

Fluxo básico:

1. Usuário seleciona o menu *Features;*
2. Usuário seleciona o menu *New Requirements;*
3. Sistema apresenta janela solicitando o número de requisitos do projeto;
4. Usuário preenche o número de requisitos e pressiona o botão *OK*;
5. Sistema apresenta uma tabela para que o usuário informe o nome dos requisitos, as fases de cada um deles e o tipo (*Functional* ou *Non Functional*);
6. Usuário preenche todos os dados da tabela e pressiona o botão *OK*;
7. Sistema apresenta a tela de Comparação de importância;

Fluxos alternativos:

3.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando ao usuário que o mesmo deve ser importado e avaliado.

4.1. Usuário preenche um dado inválido;
4.2. Sistema emite uma mensagem solicitando novamente o valor;
4.3. Sistema apresenta tela para o usuário preencher o número de requisitos;

6.1 Usuário não preenche todos os campos;

6.2 Sistema emite uma mensagem de alerta solicitando que todos os dados sejam informados.

Pré-condições: UC1.
Pós-condições: UC3.

Por sua vez, a Tabela 22 apresenta o caso de uso Avaliar requisitos, que oferece ao usuário a definição da importância de cada um dos requisitos do projeto.

Tabela 22 : UC3 – Avaliar requisitos

Fluxo básico:

1. Sistema apresenta uma matriz com todos os requisitos cadastrados;
2. Usuário insere os valores de importância entre 2 (dois) requisitos visando ao objetivo final e pressiona o botão *Calculat;*
3. Sistema utiliza o método AHP para calcular o valor em percentual de cada um dos requisitos;
4. Sistema recupera o custo total do projeto e distribui o valor total para cada um dos requisitos de acordo com o seu valor percentual;
5. Sistema emite uma mensagem perguntando se o usuário deseja verificar o valor calculado dos requisitos;
6. Usuário pressiona o botão *OK*;
7. Sistema apresenta na tela o valor de cada um dos requisitos.

Fluxos alternativos:
2.1 Usuário não preenche todos os campos;
2.2 Sistema emite uma mensagem de alerta informando que todas as células da matriz devem ser preenchidas.

6.1 Usuário pressiona o botão *Cancel;*
6.2 Sistema fecha a tela de comparação dos requisitos;

7.1 Usuário pressiona o botão *View Cost Phases.*
7.2 Sistema apresenta a matriz com os valores de cada uma das fases a que os requisitos pertencem.

Pré-condições: UC2.
Pós-condições: requisitos e fases com os devidos valores calculados.

Além disso, a ferramenta habilita ainda que seja feita a importação de um diagrama de caso de uso representado por um arquivo XMI, sendo que os passos para a ferramenta realizar esse comportamento são apresentados na Tabela 23.

Tabela 23 : UC4 – Importar XMI

Fluxo básico:

1. Usuário seleciona o menu *New Project;*
2. Usuário seleciona o menu *Import Model (XMI);*
3. Sistema apresenta uma janela do tipo *Open File*;
4. Usuário localiza o arquivo e pressiona o botão *Open*;
5. Sistema interpreta o modelo representado nesse arquivo, associando-o ao projeto criado;

Fluxo alternativo:
3.1. Caso o projeto não tenha sido cadastrado, o sistema emite uma mensagem informando que esse projeto não foi criado.

Pré-condições: UC1.
Pós-condições: UC5.

Já a Tabela 24 descreve o fluxo básico e os alternativos, que permitem ao usuário avaliar todos os elementos que fazem parte do diagrama de caso de uso do projeto.

Tabela 24 : UC5 – Avaliar elementos

Fluxo básico:

1. Sistema apresenta o nome do elemento, o *id* e o estereótipo do mesmo;
2. Sistema valida o tipo do elemento e apresenta os pesos correspondentes a cada um dos elementos;
3. Usuário seleciona o valor do peso e pressiona o botão *Next;*
4. Sistema apresenta sucessivamente os elementos seguintes;
5. Após avaliar todos os elementos, o sistema apresenta os fatores técnicos;
6. Usuário insere os pesos para cada um dos fatores e pressiona o botão *OK;*
7. Sistema apresenta os fatores de ambiente para o usuário inserir os devidos valores;
8. Usuário insere os pesos para cada um dos fatores de ambiente e pressiona o botão *OK;*
9. Sistema utiliza o método *Use Case Point;*
10. Sistema emite uma mensagem perguntando se o usuário deseja visualizar os resultados da estimativa;
11. Usuário pressiona o botão *OK*;
12. Sistema apresenta a tela de resultados e valores do custo total do projeto;
13. Usuário pressiona o botão *Close;*
14. Sistema fecha a janela de resultados.

Fluxos alternativos:

6.1. Usuário não preenche os campos corretamente;
6.2. Sistema emite uma mensagem dizendo que todos dados devem ser informados e/ou para inserir dados válidos;

8.1. Usuário não preenche os campos corretamente;
8.2. Sistema emite uma mensagem dizendo que todos dados devem ser informados e/ou para inserir dados válidos;

11.1. Usuário pressiona o botão *Cancel;*
11.2. Sistema fecha todas as janelas.

Pré-condições: UC4
Pós-condições: projeto com o custo total calculado.

Além do mais, a ferramenta disponibiliza ao usuário a gravação dos dados gerados, oferecendo a interoperabilidade entre as diversas aplicações. Em outras palavras, esse caso de

uso permite ao usuário salvar todos os valores calculados pela ferramenta, tanto do projeto como de cada requisito que faz parte do mesmo. Sendo que a Tabela 25 apresenta o fluxo principal e o alternativo desse caso de uso.

Tabela 25 : UC6 – Salvar XML

Fluxo básico:

1. Usuário seleciona o menu *Save;*
2. Sistema emite uma mensagem solicitando a versão a ser gerada do arquivo;
3. Usuário insere o valor;
4. Sistema cria um arquivo XML com o custo total do projeto e de cada requisito;
5. Sistema concatena com o valor informado no passo 3 para criar o nome do projeto e grava-o no disco;
6. Sistema emite uma mensagem informando que os dados foram salvos com sucesso.

Fluxo alternativo:

2.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que o mesmo deve ser importado.

Pré-condições: UC5.
Pós-condições: arquivo com os resultados salvos em memória não volátil.

Na seqüência do processo, através da matriz de relacionamento, a ferramenta controla os custos dos requisitos, sendo que o fluxo básico para definição da mesma é apresentado na Tabela 26, assim como os fluxos alternativos.

Tabela 26 : UC7 – Definir matriz de relacionamento

Fluxo básico:

1. Usuário seleciona o menu *Features;*
2. Usuário seleciona o menu *Set Relationship;*
3. Sistema apresenta a matriz com todos os requisitos cadastros pelo usuário para que sejam inseridos os valores em percentual;
4. Usuário entra com os valores nas células da matriz informando o valor percentual de relacionamento entre os requisitos;
5. Usuário pressiona o botão *Set;*
6. Sistema cria um grafo com a relação entre os requisitos cadastrados no sistema;
7. Usuário clica no botão *Close;*
8. Sistema fecha a janela.

Fluxo alternativos:

3.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que isso deve ser feito.

6.1. Caso o usuário insira um valor errado, o sistema emite uma mensagem exigindo que todos os dados sejam preenchidos.

Pré-condições: UC2.
Pós-condições: matriz de relacionamento cadastrada.

Após a definição da matriz de relacionamento, a ferramenta possibilita ao usuário a atualização dos valores dos requisitos, sendo que esse comportamento é descrito na Tabela 27.

Tabela 27 : UC8 – Atualizar valores dos requisitos

Fluxo básico:

1. Usuário seleciona o menu *Features;*
2. Usuário seleciona o menu *Update Requirement Values;*
3. Sistema apresenta os requisitos cadastrados no sistema com os devidos valores calculados anteriormente;
4. Usuário altera o valor de um requisito à sua escolha e pressiona o botão *Set;*
5. Sistema identifica o requisito que foi alterado;
6. Sistema consulta a matriz de valores de relacionamento entre os requisitos e atualiza o valor de todos a que ele influencia;
7. Sistema apresenta a janela com os valores alterados;
8. Usuário pressiona o botão *Close;*

Fluxo alternativo:
3.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que isso deve ser feito.

Pré-condições: UC7.
Pós-condições: valores dos requisitos atualizados, assim como também os das fases dos requisitos e do projeto.

Por fim, apresenta-se na Tabela 28 a análise dos resultados finais gerados pela ferramenta.

Tabela 28 : UC9 – Analisar resultados finais

Fluxo básico:

1. Usuário seleciona o menu *Results;*
2. Usuário seleciona o menu *Total Project;*
3. Sistema apresenta o valor do custo do projeto, o UCP, o UUCP, o valor dos fatores de ambiente e o valor dos técnicos;
4. Usuário pressiona o botão *Close;*
5. Sistema fecha a janela;
6. Usuário seleciona o menu *Results;*
7. Usuário seleciona o menu *Requirements;*
8. Sistema apresenta o valor de todos os requisitos que fazem parte do sistema;
9. Usuário pressiona o botão *Close;*
10. Sistema fecha a janela;
11. Usuário seleciona o menu *Results;*
12. Usuário seleciona o menu *Phases;*
13. Sistema apresenta o valor de todas as fases que os requisitos cadastrados integram;

14. Usuário pressiona o botão *Close;*

Fluxos alternativos:
3.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que isso deve ser feito.
8.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que isso deve ser feito.
13.1. Caso o modelo não tenha sido importado, o sistema emite uma mensagem informando que isso deve ser feito.

Pré-Condições: UC5.
Pós-Condições: -

Finalmente identificado o comportamento do sistema, o qual foi apresentado através dos diagramas de casos de uso, apresenta-se, a seguir, o diagrama de classes da ferramenta.

5.2 DIAGRAMA DE CLASSES

Para que se tenha uma visão do diagrama de classes da ferramenta, observa-se na Figura 13 a sua representação.

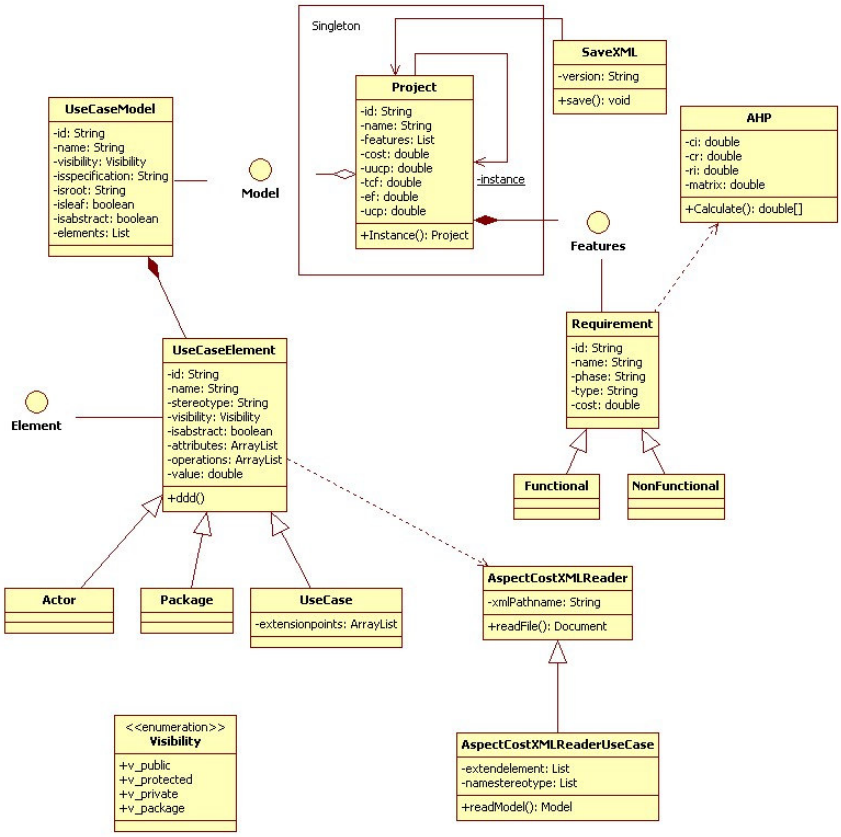

Figura 13 : Diagrama de Classes

A classe *Project* utiliza o padrão de projeto Singleton, o qual consiste em uma classe que tem um único objeto, o qual é constituído por um recurso global para todos os clientes (Gamma, 2000). Então, o contexto do padrão consiste em:

- Todos os clientes necessitam compartilhar uma única instância de uma classe;
- Deseja-se garantir que nenhuma instância adicional será criada acidentalmente.

Portanto, a solução para esse contexto é:

- Define-se uma classe com um construtor privado;
- O construtor da classe cria uma única instância de si mesmo;
- Fornece-se um método estático que retorne uma referência para a instância única.

A classe AHP é responsável por realizar os cálculos do processo *Analytical Hierarchy Process,* já a classe AspectCostXMLReader interpreta o XMI que contém o modelo do projeto (*parse* XML) e a classe SaveXML é responsável pela camada de persistência.

5.3 INTERPRETAÇÃO DO MODELO E AVALIAÇÃO DOS ELEMENTOS

Uma vez que o usuário da metodologia utiliza de uma ferramenta como *StarUML* (StarUML, 2006) e Rational Rose (Rational, 2006), por exemplo, para representar o diagrama de caso de uso do projeto, a ferramenta tem como potencialidade, a possibilidade de gerar um arquivo XMI representando esse diagrama. Então, após a geração desse arquivo XMI, o usuário importa o arquivo na ferramenta AspectCostTool. Para que se possa observar isso melhor, a Figura 14 apresenta um fragmento de um arquivo XMI de um diagrama de caso de uso.

```
13  - <XMI.content>
14  - <UML:Model xmi.id="UMLProject.1">
15  -  <UML:Namespace.ownedElement>
16  -   <UML:Model xmi.id="UMLModel.2" name="Use Case Model" visibility="public" isSpecification="false" namespace="UMLProject.1" isRoot="false" isLeaf=
17  -    <UML:Namespace.ownedElement>
18        <UML:Actor xmi.id="UMLActor.3" name="UsuÃirio" visibility="public" isSpecification="false" namespace="UMLModel.2" isRoot="false" isLeaf="false"
19        <UML:UseCase xmi.id="UMLUseCase.4" name="Checar Senha" visibility="public" isSpecification="false" namespace="UMLModel.2" clientDependency
20        <UML:UseCase xmi.id="UMLUseCase.5" name="Realizar TransaÃ§Ã£o BancÃiria" visibility="public" isSpecification="false" namespace="UMLModel.2"
21        <UML:UseCase xmi.id="UMLUseCase.6" name="Gravar InformaÃ§Ãµes de TransaÃ§Ãµes" visibility="public" isSpecification="false" namespace="UML
22        <UML:UseCase xmi.id="UMLUseCase.7" name="Visualizar Extrato de Conta" visibility="public" isSpecification="false" namespace="UMLModel.2" isRo
23        <UML:UseCase xmi.id="UMLUseCase.8" name="Visualizar Saldo de Conta" visibility="public" isSpecification="false" namespace="UMLModel.2" isRoot
24        <UML:UseCase xmi.id="UMLUseCase.9" name="Realizar TransaÃ§Ã£o de Consulta" visibility="public" isSpecification="false" namespace="UMLModel
25        <UML:UseCase xmi.id="UMLUseCase.10" name="Cobrar Tarifa por TransaÃ§Ã£o Extra" visibility="public" isSpecification="false" namespace="UMLM
26  -     <UML:Association xmi.id="UMLAssociation.11" name="" visibility="public" isSpecification="false" namespace="UMLModel.2">
27  -      <UML:Association.connection>
28          <UML:AssociationEnd xmi.id="UMLAssociationEnd.12" name="" visibility="public" isSpecification="false" isNavigable="false" ordering="unordered"
29          <UML:AssociationEnd xmi.id="UMLAssociationEnd.13" name="" visibility="public" isSpecification="false" isNavigable="true" ordering="unordered"
30        </UML:Association.connection>
31      </UML:Association>
32  -     <UML:Association xmi.id="UMLAssociation.14" name="" visibility="public" isSpecification="false" namespace="UMLModel.2">
33  -      <UML:Association.connection>
34          <UML:AssociationEnd xmi.id="UMLAssociationEnd.15" name="" visibility="public" isSpecification="false" isNavigable="false" ordering="unordered"
35          <UML:AssociationEnd xmi.id="UMLAssociationEnd.16" name="" visibility="public" isSpecification="false" isNavigable="true" ordering="unordered"
36        </UML:Association.connection>
37      </UML:Association>
38      <UML:Dependency xmi.id="UMLDependency.17" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" client="UMLUseCase.1
39      <UML:Dependency xmi.id="UMLDependency.18" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" client="UMLUseCase.4
40      <UML:Dependency xmi.id="UMLDependency.19" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" client="UMLUseCase.6
41      <UML:Dependency xmi.id="UMLDependency.20" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" client="UMLUseCase.6
42      <UML:Generalization xmi.id="UMLGeneralization.21" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" discriminator="" ch
43      <UML:Generalization xmi.id="UMLGeneralization.22" name="" visibility="public" isSpecification="false" namespace="UMLModel.2" discriminator="" ch
44      <UML:Stereotype xmi.id="X.27" name="useCaseModel" extendedElement="UMLModel.2"/>
45      <UML:Stereotype xmi.id="X.28" name="Aspectual" extendedElement="UMLUseCase.4 UMLUseCase.6 UMLUseCase.10"/>
46    </UML:Namespace.ownedElement>
47   </UML:Model>
```

Figura 14 : Fragmento de arquivo XMI

Observa-se nas linhas de 19 a 25 que as *tags* <UML:UseCase> representam os casos de uso do diagrama. Por sua vez, a linha 18, retratada pela tag <UML:Actor>, representa os atores envolvidos no comportamento do sistema. E, por fim, as linhas 44 e 45 reproduzem os estereótipos dos elementos do diagrama que a ferramenta necessita interpretar para identificar

se os mesmos são aspectuais ou não. Sendo assim, o diagrama de seqüência, conforme pode-se notar na Figura 15, expõe o comportamento dinâmico da AspectCostTool para interpretar esse arquivo.

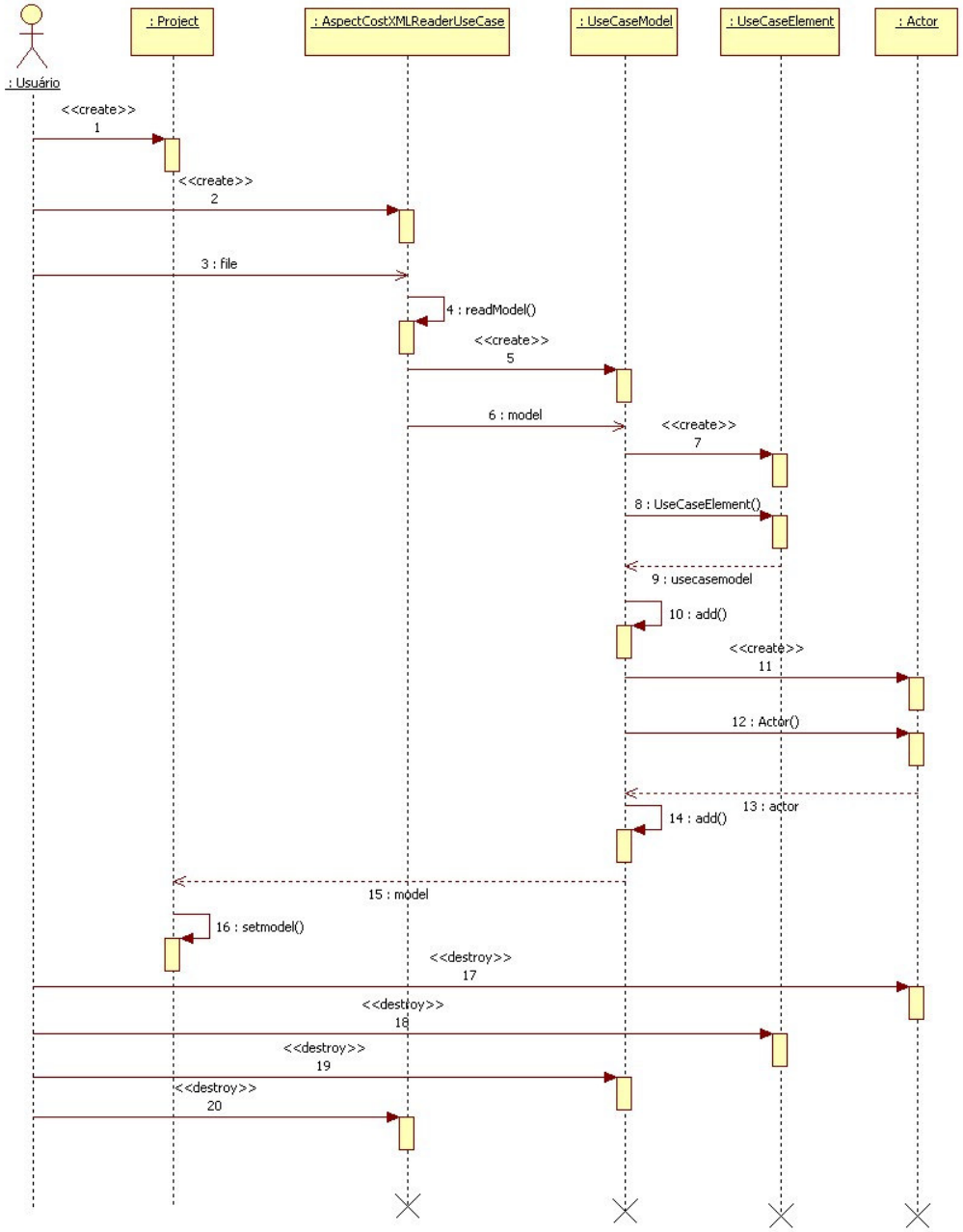

Figura 15 : Diagrama de seqüência – Ler Arquivo

O método *readModel()* invocado pelo objeto da classe AspectCostXMLReaderUseCase utilizou-se do DOM (W3C, 2003) para executar o *parse* do

modelo. A ferramenta carrega esse arquivo e identifica todos os atores, casos de uso, casos de uso aspectual e seus devidos estereótipos, instanciando cada um dos objetos necessários, faz os devidos relacionamentos e, por fim, instancia um objeto do tipo *Model*, associando-o ao projeto criado pelo usuário.

5.4 AVALIAÇÃO DOS ELEMENTOS

Com o modelo instanciado na ferramenta, são fornecidos os pesos dos elementos levando em consideração o novo elemento representado pelo caso de uso aspectual. Portanto, salienta-se que para cada elemento, os pesos corretos são fornecidos como parâmetros de entrada ao usuário. Dessa forma, a Figura 16 exemplifica a avaliação de um ator no diagrama de caso de uso, e apresenta os valores propostos pela metodologia de estimativa *Use Case Point.*

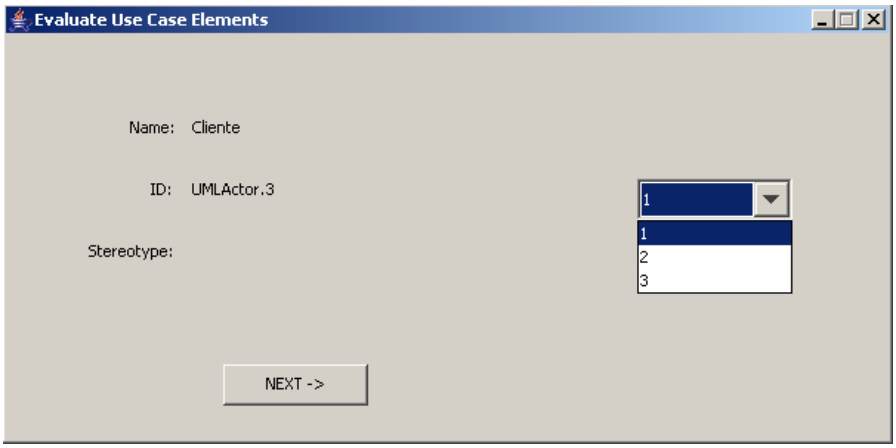

Figura 16 : Avaliar ator

A Figura 17 apresenta a avaliação de um caso de uso integrante do diagrama, cuja identificação é realizada através do estereótipo do mesmo, considerando que os pesos referem-se ao número do passo de cada elemento.

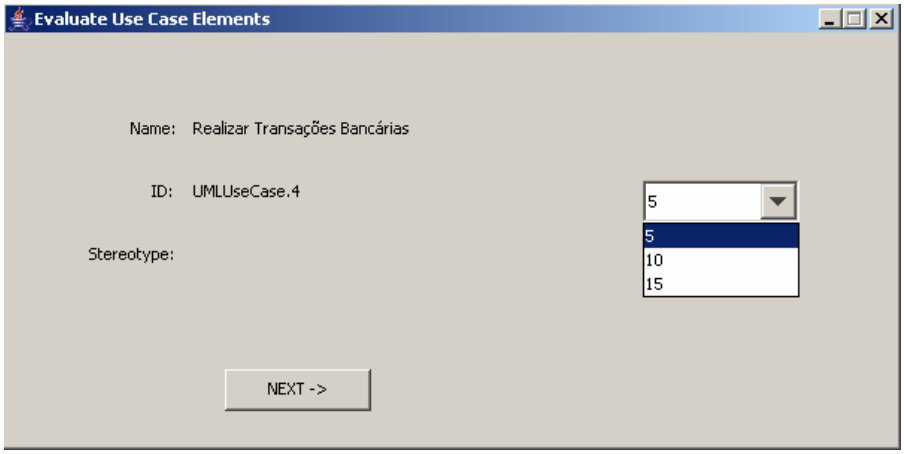

Figura 17 : Avaliar casos de uso

Já a Figura 18 apresenta a avaliação de um caso de uso aspectual identificado através de seu estereótipo, conforme exemplo, sendo que esse elemento é identificado através do arquivo XMI apresentado na Figura 14. A ferramenta apresenta ainda as telas para todos os elementos que necessitam ser avaliados, a fim de realizar o cálculo do projeto.

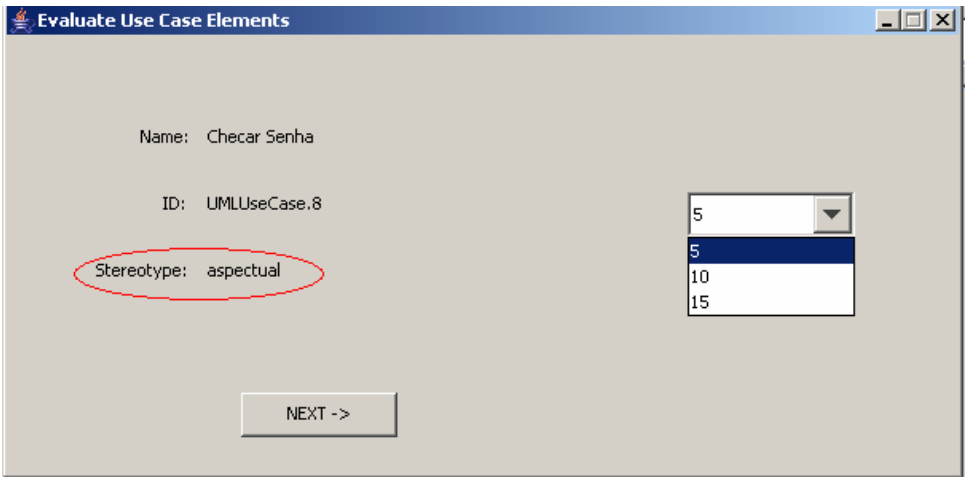

Figura 18 : Avaliar caso de uso aspectual

Após a avaliação dos elementos do diagrama de caso de uso, a ferramenta fornece os parâmetros de entrada para o usuário inserir os fatores técnicos e de ambiente, sendo que o resultado desse processo é o custo total do projeto, conforme mostra a Figura 19.

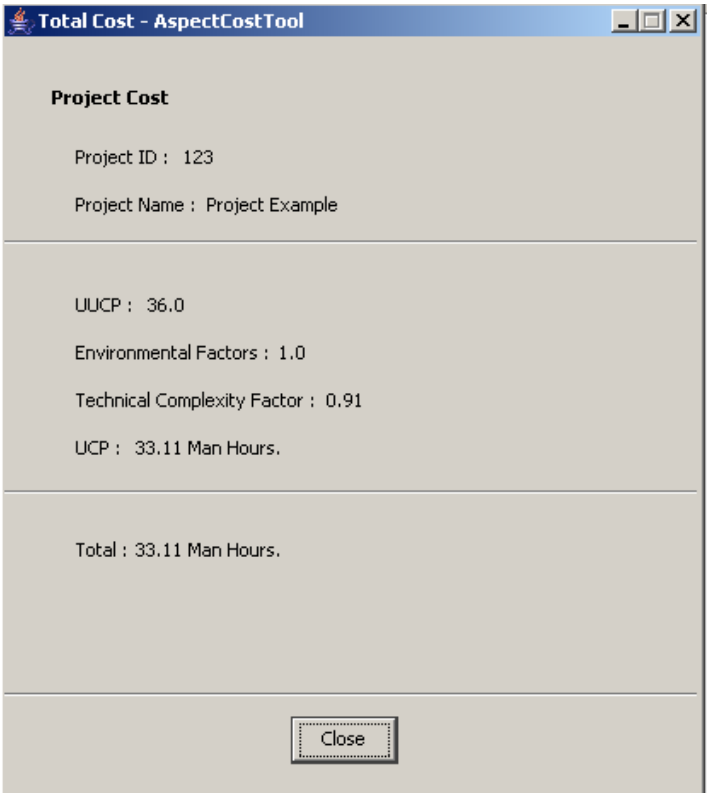

Figura 19 : Resultado total do projeto

A Figura 19 mostra que o projeto tem um custo de 33.11 homens/horas. A partir deste valor é possível cadastrar os requisitos do projeto e, posteriormente, identificar o custo de cada um deles.

5.5 CADASTRO DOS REQUISITOS E CUSTOS DOS REQUISITOS

A ferramenta disponibiliza ainda a inserção de todos os requisitos que fazem parte do projeto, conforme mostra a Figura 20, sendo que as opções disponíveis no momento do cadastro são as seguintes:

- Nome do requisito: breve descrição da funcionalidade, o que facilita a identificação do objetivo.

- Fase do requisito: caso o processo de desenvolvimento esteja baseado em um modelo incremental, como RUP por exemplo, disponibiliza-se o relacionamento dos requisitos com as fases do projeto;

- Tipo: identifica-se se o requisito é funcional ou não, no entanto esse campo é meramente informativo, e, portanto, não altera nenhum comportamento da ferramenta.

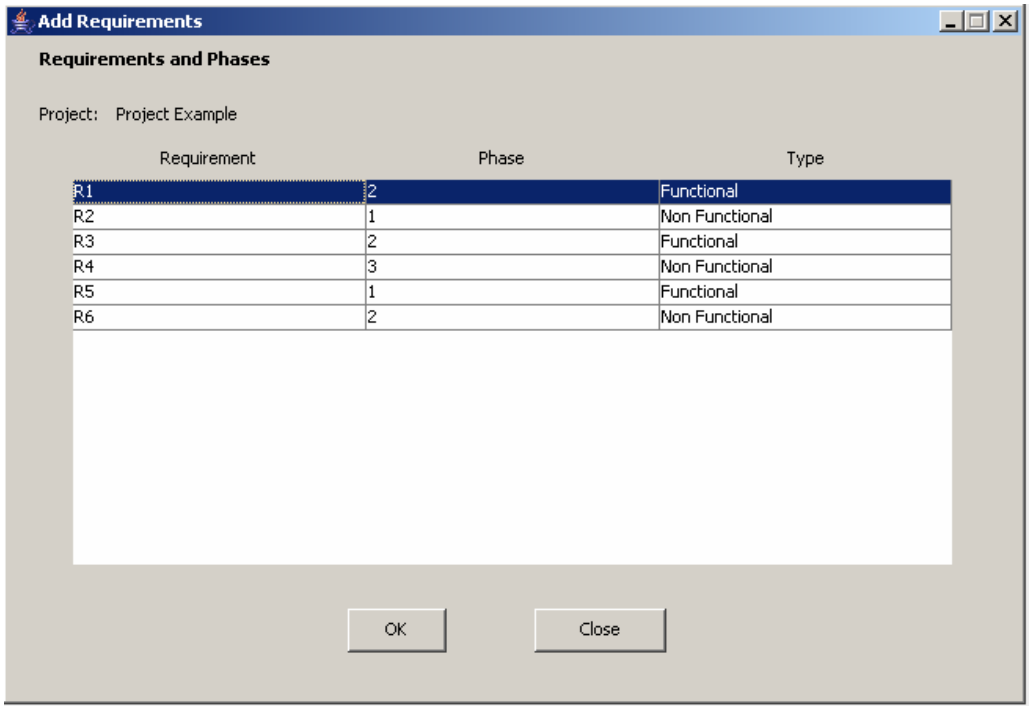

Figura 20 : Cadastro dos Requisitos

Portanto, concluindo a etapa de cadastro, a ferramenta habilita a matriz de importância para que sejam informados os dados pré-requisitos para o cálculo AHP, como pode-se observar na Figura 21. Então após a realização do processo AHP, a ferramenta divide o valor do projeto entre os valores percentuais de cada um dos requisitos, e o resultado dessa etapa é o custo de cada um dos requisitos.

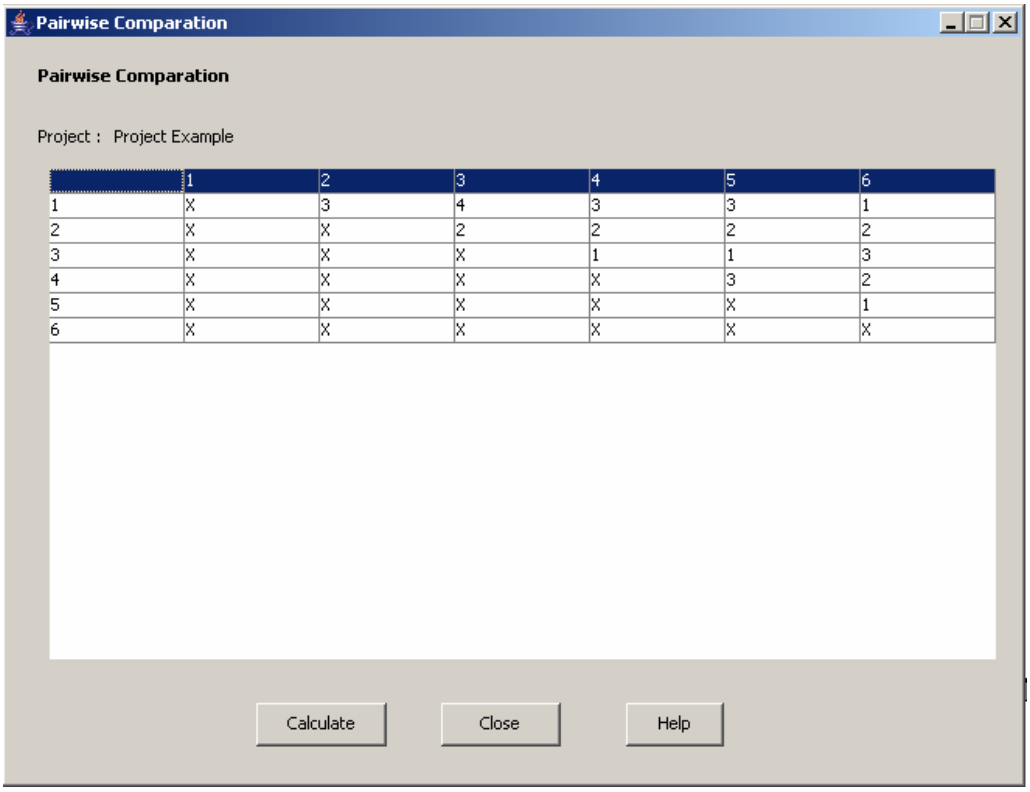

Figura 21 : Matriz de comparação

Após a inserção dos valores na matriz de comparação, o sistema realiza o cálculo do processo AHP desenvolvido na classe AHP, sendo que a Figura 22 apresenta o comportamento dinâmico desse processo.

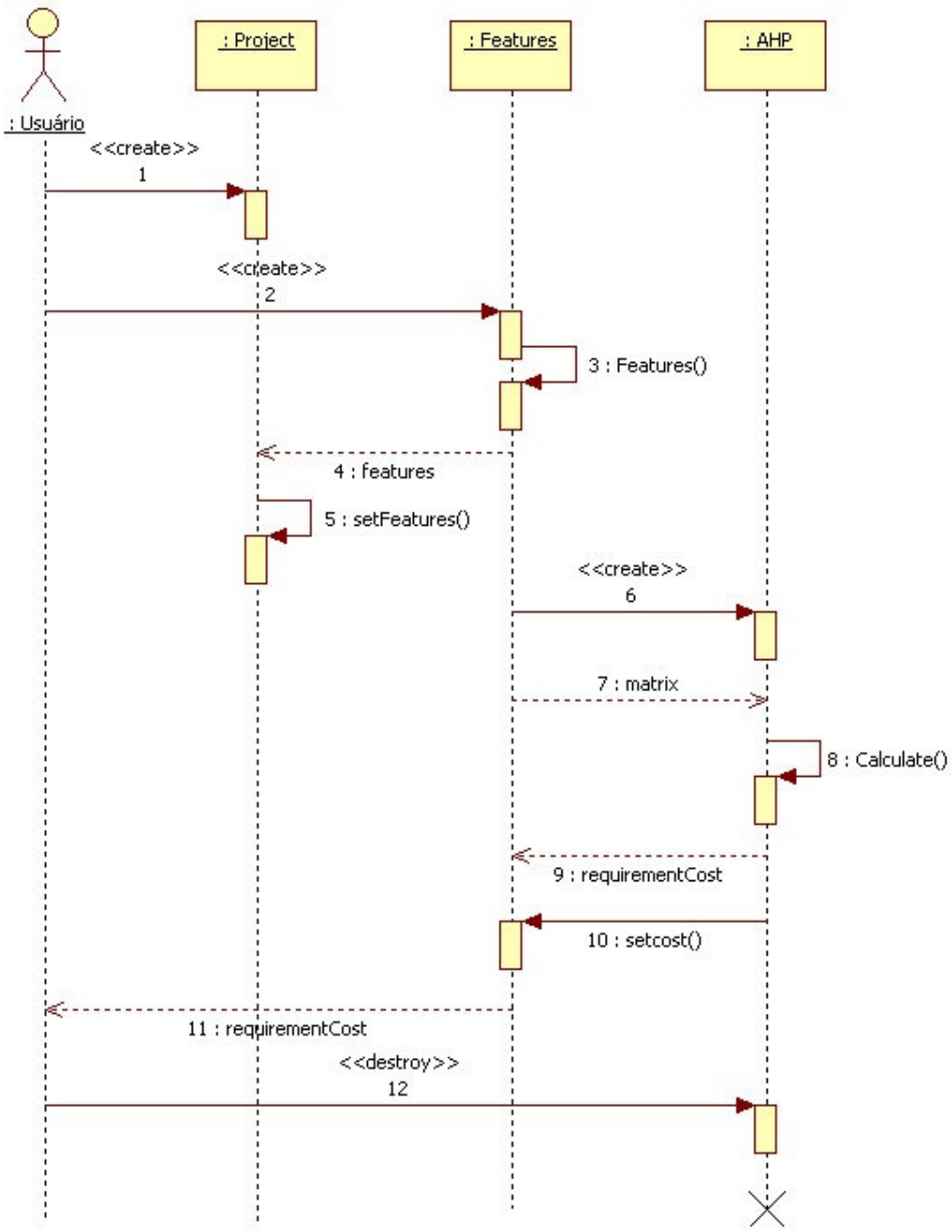

Figura 22 : Diagrama de seqüência – Calcular custo dos requisitos

Após a chamada do método *Calculate()* da classe AHP, os valores, em percentual, de cada um dos requisitos são associados aos mesmos, sendo que o resultado desse processo é apresentado na Figura 23, na qual observa-se que cada requisito do projeto detêm um custo. Então, é dada a flexibilidade ao usuário do cálculo na identificação dos maiores e menores

custos, sendo que o mesmo pode identificar os recursos e esforços necessários para a construção de cada um deles.

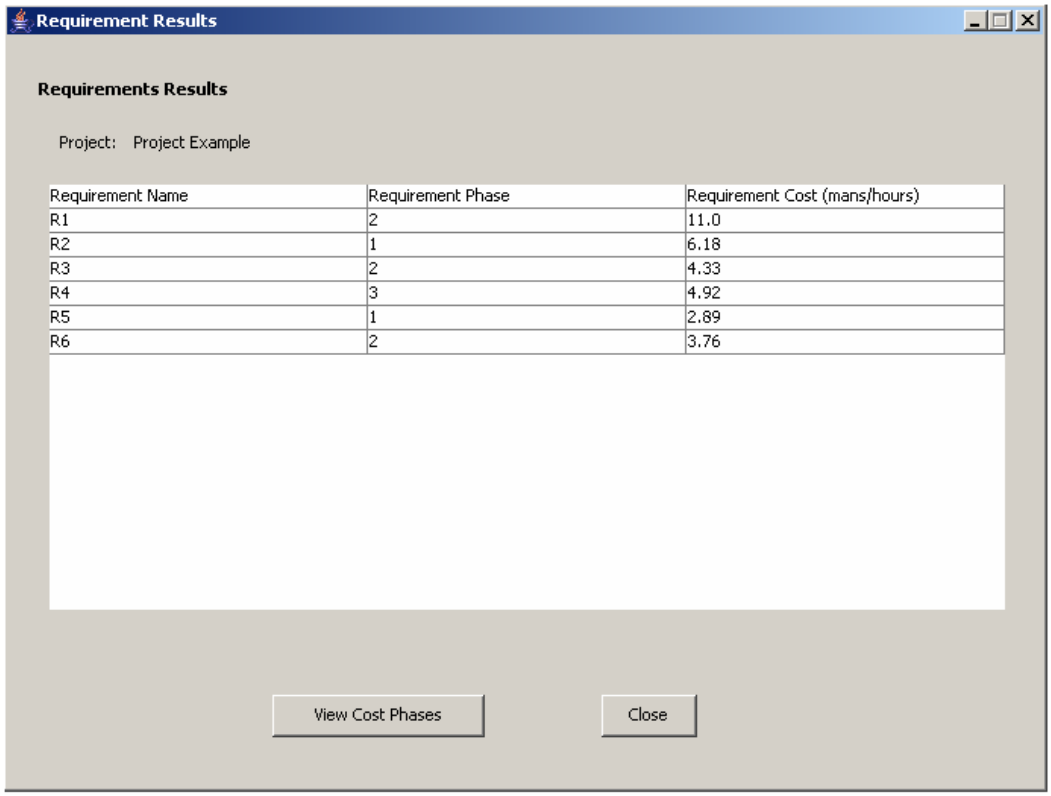

Figura 23 : Resultado Custos dos Requisitos

A ferramenta ainda disponibiliza o valor de cada uma das fases dos requisitos, caso o projeto utilize o *framework* RUP para auxiliar no desenvolvimento. Portanto, considerando todas as contribuições da ferramenta, destaca-se a maior delas, ou seja, o mapeamento dos valores ao longo do projeto, o qual é apresentado na seção 5.6.

5.6 MAPEAMENTO DOS VALORES DOS REQUISITOS

Os requisitos não estão isolados em um projeto, pois muitos se relacionam entre si, causando o efeito de encadeamento entre eles. Sendo assim, considerando que um *software* é a agregação de vários requisitos, é possível concluir que os seus custos impactam diretamente no custo total do projeto. Porém, uma das grandes dificuldades na área de Engenharia de Requisitos é a gerência de configuração, que trata especificamente das alterações dos mesmos

ao longo do ciclo de vida de um projeto, sendo que tais alterações podem representar um novo processo que o cliente necessita ou até mesmo uma modificação nos requisitos já especificados.

Portanto, as mudanças de escopo dos requisitos causam um impacto no projeto caso não seja feita uma análise de impacto e, até mesmo a análise de risco. Isso significa que as alterações nos requisitos impactam diretamente o custo, provendo um aumento no valor, e conseqüentemente, no total do projeto. No entanto, identificado o problema, a ferramenta AspectCostTool provê o mapeamento dos valores em todas as camadas que se relacionam com os requisitos, sejam nas fases às quais eles pertencem ou até no custo total do projeto.

Sendo assim, o valor do parâmetro a ser adicionado ou subtraído de um requisito está diretamente relacionado com a matriz de relacionamento cadastrada pelo usuário da ferramenta. Então, a Figura 24 apresenta a característica da ferramenta que mantém este controle.

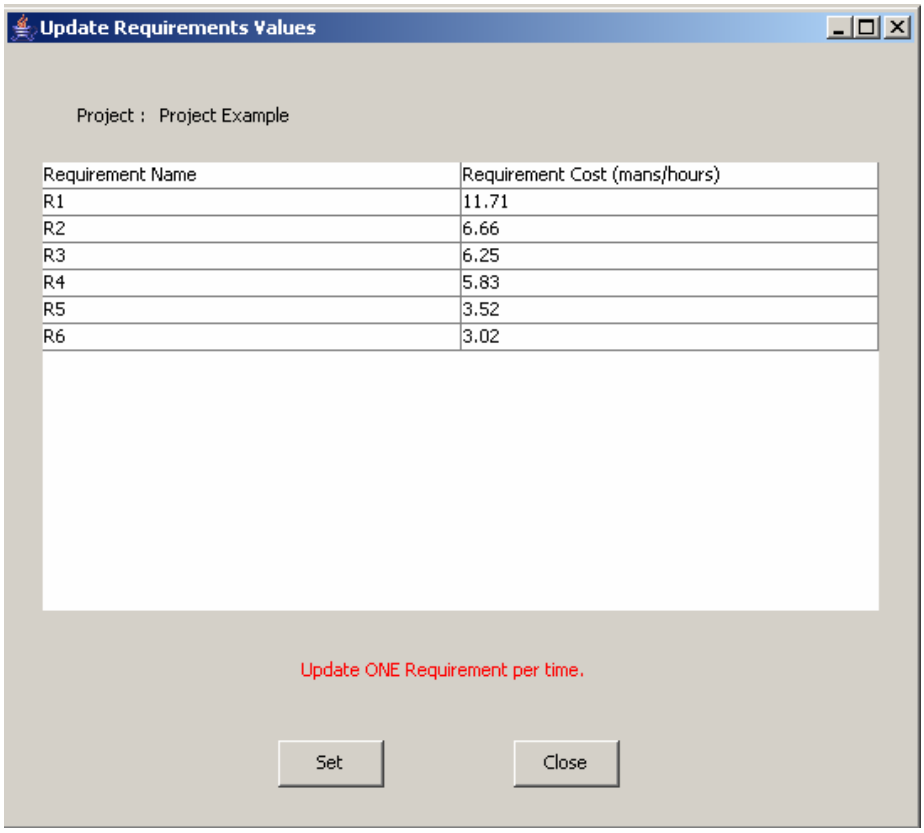

Figura 24 : Atualização do custo do requisito

Pode-se notar então que o exemplo acima demonstra que cada um dos custos possui valores. Assim, simulando a alteração de um requisito, como por exemplo o requisito R1 que detêm de 11.71 homens/horas para 14, seja por um novo artefato que fará parte do mesmo, a ferramenta mapeia a atualização ao longo dos demais.

Após essa identificação, o sistema realiza o procedimento apresentado na seção 4.4 deste documento e aponta todos os novos valores. Dessa forma, feita a atualização de todos os valores, a ferramenta soma-os e associando-os ao custo total do projeto, provendo, assim, a visibilidade de qualquer tipo de alteração. Além do mais, a grande vantagem desse controle é oferecer ao cliente a visão de quanto uma simples alteração em decorrência da sua necessidade impacta ao longo do projeto, o que contribui para a tomada de decisão sobre a aceitação ou não de um requisito.

Finalmente, considerando que o cliente ou mesmo a companhia que esteja desenvolvendo a solução utilize outra ferramenta de gerenciamento de projetos, a ferramenta AspectCostTool grava essas informações em um arquivo XML, provendo, dessa forma, a interoperabilidade entre os dados. Portanto, o ganho maior dessa característica, a qual é apresentada na seção 5.7, é não utilizar a ferramenta AspectCostTool de forma isolada das demais envolvidas no processo de desenvolvimento.

5.7 CAMADA DE PERSISTÊNCIA (XML)

A camada de persistência da ferramenta AspectCostTool foi desenvolvida sobre o padrão XML, sendo que o maior benefício em utilizar esse tipo de estrutura, como já foi mencionado, é a interoperabilidade entre as aplicações. Além disso, a ferramenta que porventura desejar utilizar-se dos dados gerados pela AspectCostTool deve apenas adicionar um *parser* (interpretador) à estrutura criada e gerada. Para exemplificar essa camada, A Figura 25 apresenta um fragmento de um arquivo XML gerado pela ferramenta.

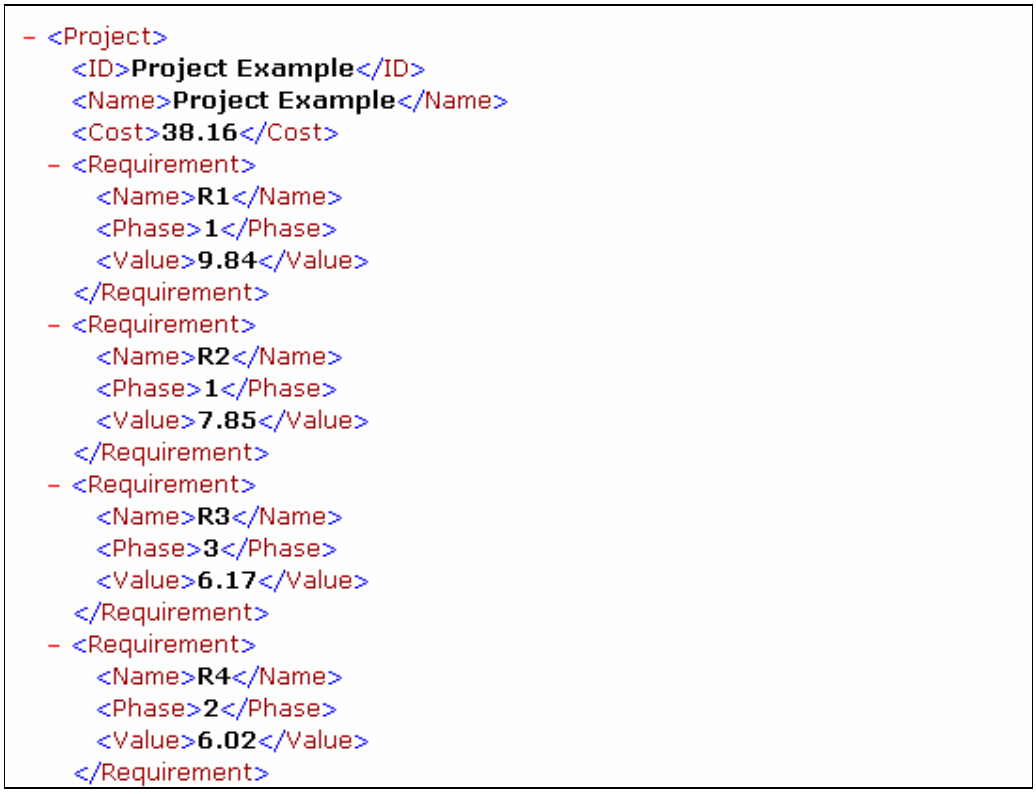

Figura 25 : Fragmento de um Arquivo XML

Portanto, nota-se que os dados gerados pela ferramenta são:

- Dados do projeto: consiste em dados informativos, como nome, identificador e o custo total;

- Dados dos requisitos e fases: todos os dados dos requisitos, considerando seu nome, a fase a que pertence e o custo de cada um deles.

Então, visando à validação da metodologia e da ferramenta aqui apresentadas, utilizou-se de um sistema exemplo que considera os artefatos de implementação orientada a aspecto para realizar um estudo de caso, o qual é apresentado na seção 6.

6. ESTUDO DE CASO

O estudo de caso aqui apresentado preocupa-se com a avaliação dos benefícios da metodologia AspectCost e se ela resulta no esperado. Assim, esse estudo de caso tem o propósito de:

- Demonstrar como a metodologia AspectCost pode ser utilizada para estimar o custo de projetos que utilizam o paradigma de desenvolvimento orientado a aspecto;
- Avaliar se a proposta está coerente com os cálculos dos processos utilizados ao longo da metodologia em um sistema real, mensurando requisitos e controlando-os.

Sendo assim, para realizar o estudo de caso foi utilizado um sistema bancário disponível na Internet, também denominado de *Internet Banking* (Souza, 2004). O motivo pelo qual esse foi escolhido é o fato de ele possuir todos os artefatos necessários para a utilização da metodologia AspectCost, tais como: casos de uso aspectuais, casos de uso, atores e os devidos requisitos; uma vez que o principal objetivo do sistema é fornecer para os clientes a possibilidade de realizar transações bancárias *online*. Assim, a Figura 26 apresenta o comportamento do sistema especificado através do diagrama de caso de uso.

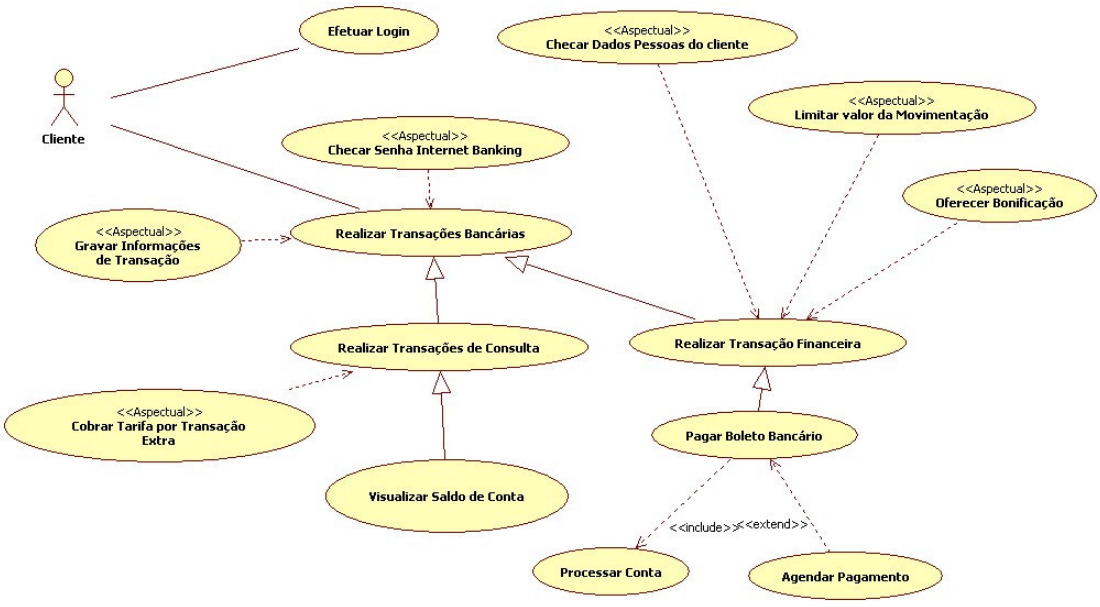

Figura 26 : Diagrama de caso de uso do *Internet Banking*

O sistema consiste na possibilidade de realização de transações bancárias via Internet. Os aspectos do sistema encortam as principais transações da aplicação, sendo as Transações Bancárias que possui a maioria das funcionalidades e dissemina para os casos de uso filho, neste caso, Realiza Transações de Consulta e Visualizar Saldo de conta. O aspecto Oferece Bonificação apenas influencia as Transações financeiras com o relacionamento de dependência, com isso, no momento que é executa este tipo de transação o aspecto afeta. O aspecto Checar Senha Internet Banking afeta todas as funcionalidades da aplicação, sendo que a cada ação do ator Cliente o aspecto irá atuar.

Por sua vez, os requisitos do sistema são apresentados na Tabela 29:

Tabela 29 : Requisitos do sistema do *Internet Banking*

Nome	**Tipo**
Efetuar *login*	Funcional
Realizar transações bancárias	Funcional
Realizar transações de consulta	Funcional
Visualizar saldo de conta	Funcional
Realizar transações financeiras	Funcional
Pagar boleto bancário	Funcional
Processar conta	Funcional
Agendar pagamento	Funcional
Gravar informações de transação	Aspectual
Cobrar tarifa por transação extra	Aspectual
Checar senha de *Internet Banking*	Aspectual
Chegar dados pessoais do cliente	Aspectual
Limitar valor da movimentação	Aspectual
Oferecer bonificação	Aspectual

Pode-se observar então que o arquivo XMI, representando o diagrama de caso de uso do sistema, foi importado para a ferramenta AspectCostTool e todas as devidas avaliações dos elementos baseado nos critérios estabelecidos, tanto para atores, como para casos de uso e

casos de uso aspectuais. Os valores dos pesos dos elementos não são importados através do arquivo XML, sendo que devem ser digitados pelo usuário da ferramenta. A Figura 27 apresenta o custo total do projeto previamente calculado:

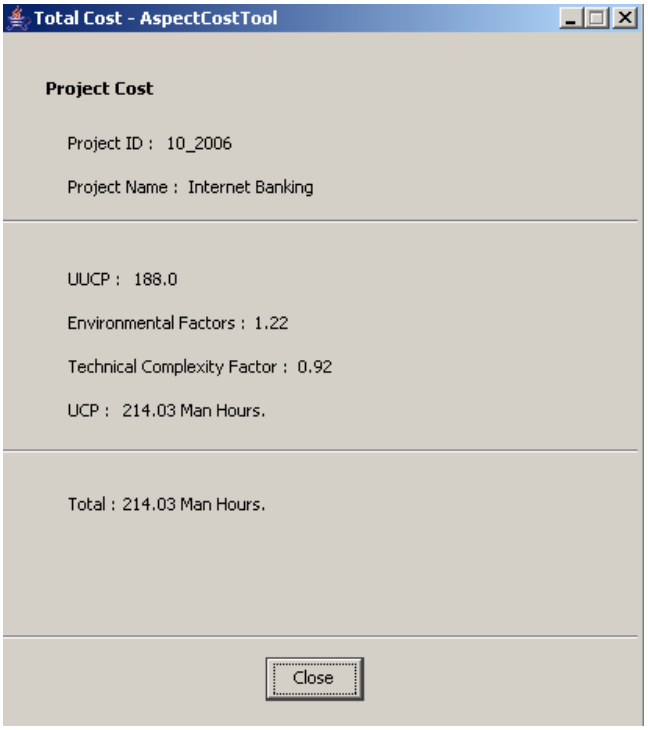

Figura 27 : Custo total *Internet Banking*

Além disso, todos os requisitos do sistema foram inseridos na ferramenta conforme é visto na Figura 28, permitindo à validação da entrada de dados. No entanto, para garantir a acurácia dos cálculos implementados na ferramenta, foram subjetivamente inseridos valores para realizar o cálculo do método AHP e para a validação da implementação foram inseridos os mesmos valores em outro *software* da *Canandian Conservation Institute* (CCI).

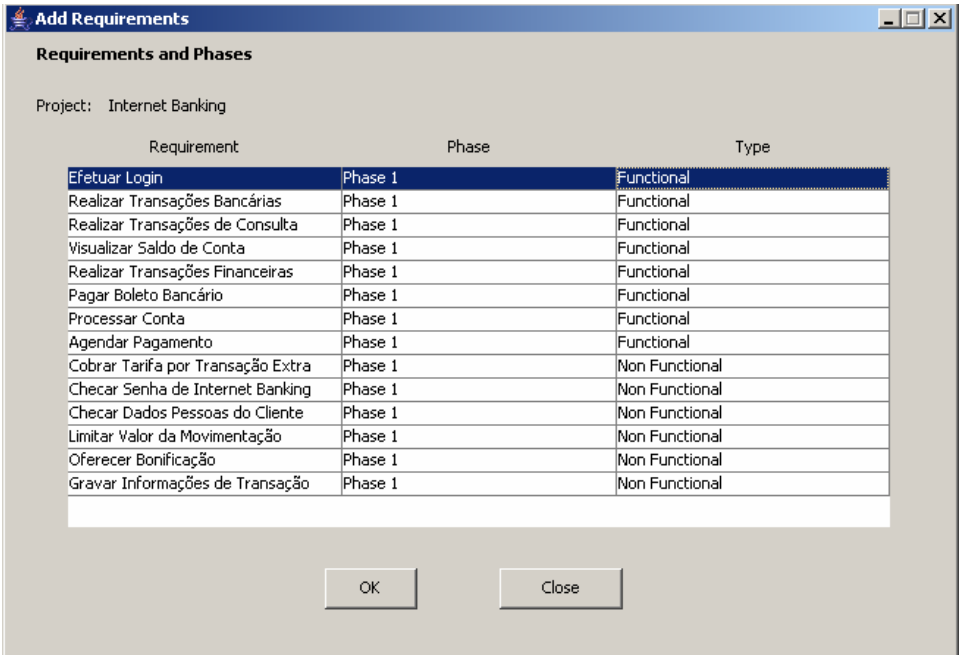

Figura 28 : Requisitos do *Internet Banking*

Ainda é importante salientar que os resultados obtidos pelo *software* do CCI foram estes, demonstrados a seguir em percentual, conforme Figura 29:

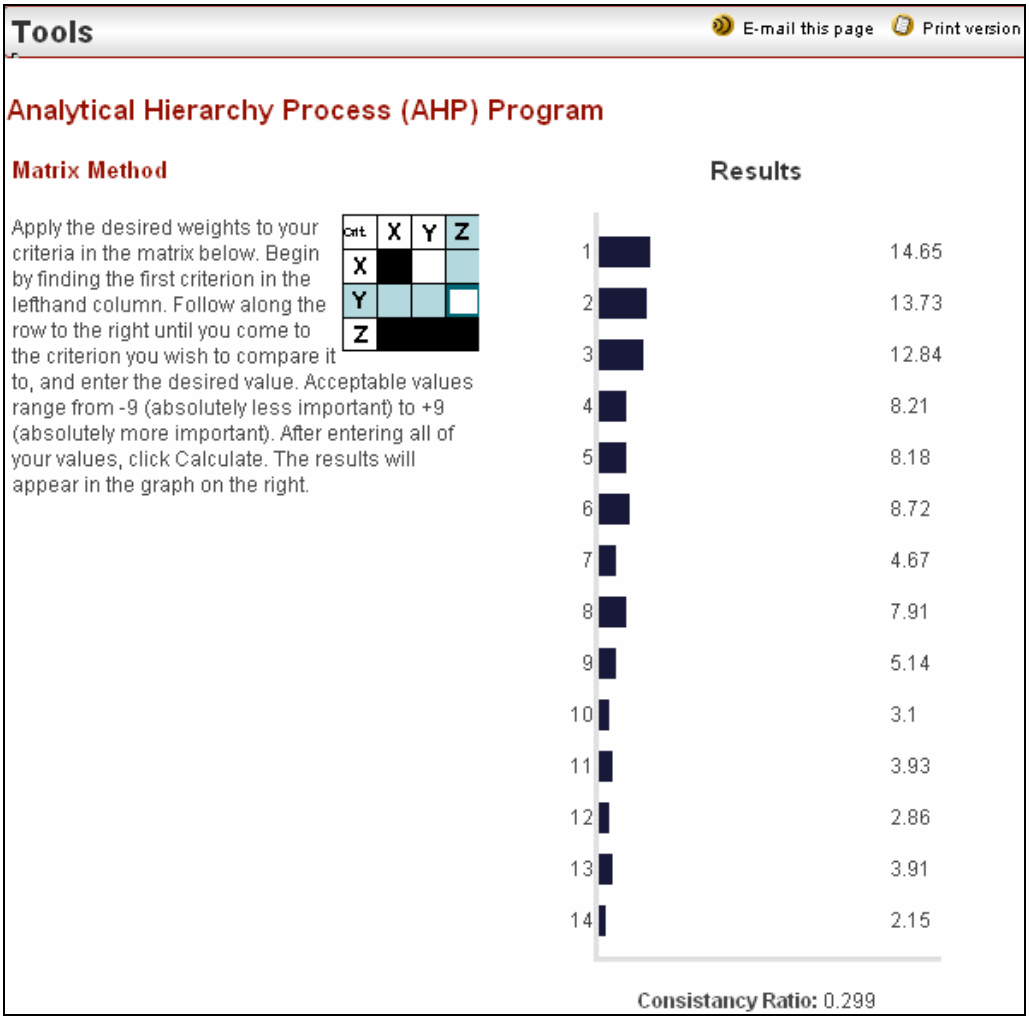

Figura 29 : Resultado do método AHP – CCI Tool

Pode-se observar, assim, que os resultados gerados pela ferramenta AspectCostTool são os valores dos custos de cada um dos requisitos, porém se considerar a porcentagem sobre o total, observa-se a confiança no cálculo implementado.

Desta forma, a Figura 30 apresenta os valores calculados e sua devida porcentagem em relação ao custo total do projeto.

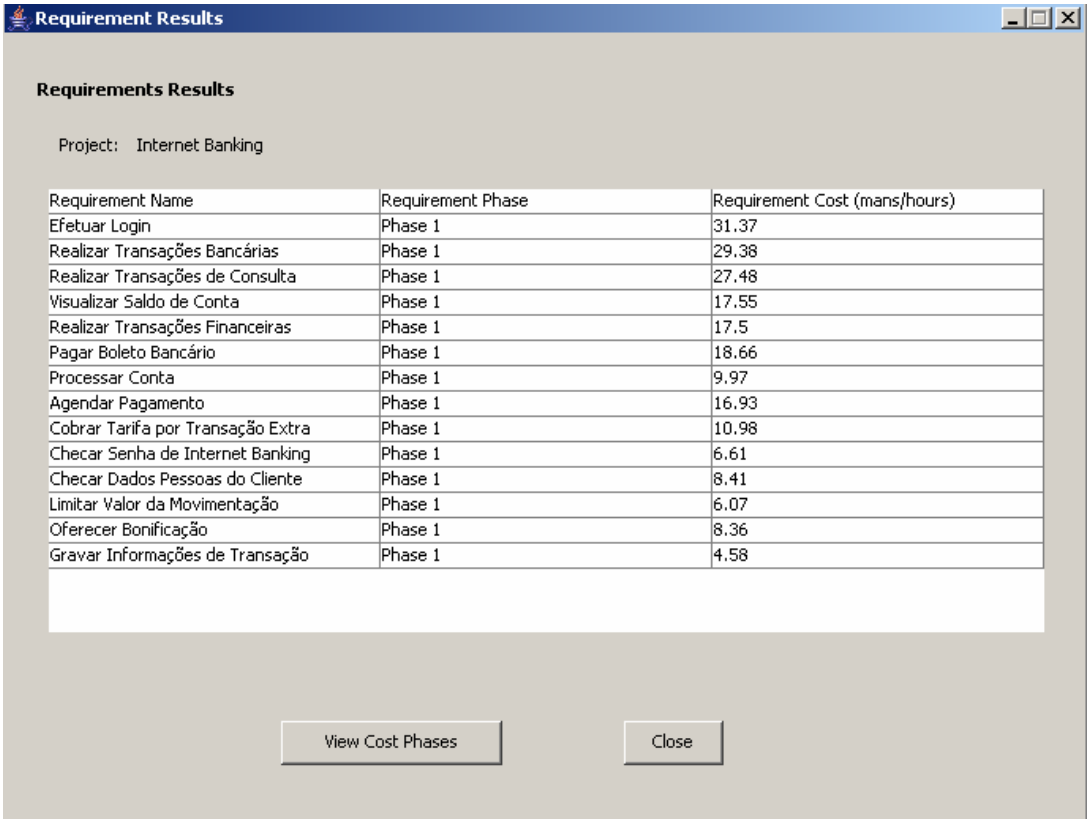

Figura 30 : Custos dos requisitos pela AspectCostTool

Por sua vez, a Tabela 30 apresenta os valores que podem ser comparados com a outra ferramenta:

Tabela 30 : Acurácia do cálculo

Requisito	Custo (homens/hora)	% AspectCostTool	% CCI Tool
Efetuar *login*	31.37	14,65 %	14,65%
Realizar transações bancárias	29.38	13.72%	13.73%
Realizar transações de consulta	27.48	12.83%	12.84%
Visualizar saldo de conta	17.55	8.19%	8.21%
Realizar transações financeiras	17.5	8.17%	8.18%
Pagar boleto bancário	18.66	8.71%	8.72%
Processar conta	9.97	4.65%	4.67%
Agendar pagamento	16.93	7.91%	7.91%
Cobrar tarifa por transação extra	10.98	5.13%	5.14%
Checar senha de *Internet Banking*	6.61	3.08%	3.1%
Checar dados pessoais do cliente	8.41	3.92%	3.93%
Limitar valor da movimentação	6.07	2.83%	2.86%
Oferecer bonificação	8.36	3.90 %	3.91%
Gravar informações de transações	4.58	2.13%	2.15%

Portanto, através desse estudo de caso, consegue-se obter os objetivos propostos, não visando à garantia total da metodologia, mas à validação dos cálculos implementados. No entanto, para a validação coerente da metodologia seria necessário fazer a sua simulação e a utilização em vários experimentos, o que é citado como possível trabalho futuro. Considerando que é possível com o presente estudo de caso responder a questão de pesquisa,

pois foi possível estimar o custo dos requisitos de um projeto que utiliza de artefatos AOP, através da adaptação do *Use Case Points* e da utilização do processo AHP. Com o algoritmo de controle, este baseado na tabela de relacionamentos, é possível controlar cada alteração dos requisitos do projeto, provendo uma visibilidade para os gestores nas tomadas de decisões.

7. CONCLUSÃO

Após a análise do presente trabalho, é possível concluir que sua principal contribuição está centrada na proposta metodológica, que propõem estimar os custos de projetos e de requisitos, adaptada ao paradigma de desenvolvimento orientado a aspecto. Além disso, a metodologia apresenta uma proposta diferenciada voltada para um dos problemas da área de Engenharia de *Software*, o qual torna-se mais crítico quando há a necessidade de valorar requisitos em AOP e de estabelecer uma melhor gerência do projeto.

Para o desenvolvimento do trabalho foram estudadas técnicas computacionais e gerenciais necessárias para implementação e disponibilização de recursos. Assim, foram realizados estudos e experimentos sobre os modelos de estimativa de custos; gerência e acompanhamento de custos; engenharia de requisitos; e gerência de configuração, ou mais especificamente, a gestão de mudanças.

Para tanto, o estudo de caso e os experimentos realizados validaram as hipóteses iniciais, as quais são:

a) Estimar custos considerando os conceitos de orientação a objeto, visto que a orientação a aspecto complementa-o;

b) Utilizar uma representação gráfica e quantitativa para representar os aspectos do sistema;

c) Utilizar estimativas baseadas em modelos de análise com a utilização da UML como linguagem de representação;

d) Utilizar um processo quantificador para atribuir custos aos requisitos e relacioná-los.

Sendo assim, como forma de validar os procedimentos da metodologia AspectCost foi realizado um estudo de caso para avaliar os passos implementados; validando dessa forma a ferramenta que acompanha a metodologia. Além disso, dentro do estudo de caso, testes de verificação e validação de dados de entrada e de saída foram efetuados com o sistema proposto na dissertação de Souza, o qual detêm todos os aspectos necessários para utilização da metodologia. Conseqüentemente, o estudo de caso teve por objetivo principal não apenas

validar completamente a estimativa, mas, também, verificar se os requisitos da metodologia são coerentes com a especificação.

Além do mais, a pesquisa ainda traz as seguintes contribuições:

- Apresentar uma ferramenta totalmente orientada a objeto que provê ao gerente de projeto a identificação do impacto de um novo requisito, ou a sua alteração, no custo total do mesmo;
- Mostrar uma ferramenta expansível para qualquer outro elemento que participe do cálculo da estimativa do projeto;
- Apresentar uma camada de interoperabilidade para servir de *input* para outros ambientes de gerência de projeto que tenham interesse em utilizar os dados gerados pela ferramenta;
- Mostrar uma metodologia totalmente adaptada ao modelo de estimativa COCOMO II, visto que se utiliza de outros para calcular o custo total do projeto;
- Apresentar uma metodologia para o controle dos custos dos requisitos e o mapeamento destes valores.

Portanto, pode-se concluir que os objetivos do trabalho foram alcançados sendo que são citados alguns aspectos futuros quem complementam a pesquisa.

7.1 Trabalhos futuros

A seguir, são apresentados alguns trabalhos que poderão ser realizados como continuidade dessa dissertação:

- Realizar mais experimentos para validar a metodologia;
- Incluir um maior número de artefatos para o cálculo do custo total do projeto, tais como, diagramas dinâmicos da UML;

- Desenvolver um *plug-in* para IDEs com Eclipse, centrando, assim, todo tipo de ferramenta disponível para o desenvolvimento de um projeto;

- Desenvolver uma camada *web* para centralizar todo e qualquer tipo de cálculo, facilitando a manutenção da aplicação.

8. REFERÊNCIAS BIBLIOGRÁFICAS

(**Almeida, 2005**) Almeida A. "**JCommerceNet: Um Framework usando aspectos na geração de scripts para camada de persistência para automação de lojas virtuais".** Universidade do Vale do Rio dos Sinos, 2005.

(**Agarwal, 2001**) Agarwal R., Kumar M., Mallick S., Bharadwaj R., Anantwar. "**Estimating Software Project**". In ACM SIGSOGT, vol 26, n. 4, 2001.

(**Boehm, 2000**) Boehm B., Horowitz E., Madachy R., Reifer D., Clark B., Steece B., Brown A., Chulani S., Abts C. "**Software Cost Estimation with COCOMO II**". Prentice Hall, 2000.

(**Booch, 2000**)Booch G., Rumbaugh J., Jacobson I. "**A Managing Software Requirements**". Indianapolis: Addison-Wesley, 2000.

(**Booch, 2000**) Booch G., Rumbaugh J., Jacobson I. "**UML – Guia do Usuário**". Rio de Janeiro: Elsevier. 13ª reeimpressão, 2000.

(**Carbone, 2002**) Carbone M., Santucci G. "**Fast&&Serious: A UML based metric for effort estimation**". In 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering. Málaga, Spain, 2002.

(**Chitchyan, 2005**) Chitchyan, R.; Rashid A.; Sawyer P.; Garcia A.; Alarcon M.; Bakker J.; Tekinerdogan B.; Clarke S.; Jackson A. "**Survey of Aspect-Oriented Analysis and Design Approaches".** In AOSD-EUROPE., 2005.

(**Elrad, 2001**) Elrad T., Filman R., Bader A. "**Discussing Aspects of AOP**". In Communication ACM, vol. 44, n. 10, 2001.

(**Fox, 2005**) Fox T., Spence J., "**The Effect of Decision Style on the Use of a Project Management Tool: An Empirical Laboratory Study**". In ACM, The DATA BASE for Advances in Information Systems – Spring 2005. Vol. 36. N. 2.

(**Gamma, 2006)** Gamma, E., Helm R., Johnson R. "**Padrões de Projeto"**. Bookman, 2000.

(**Gutierrez, 2003**) Gutierrez J. F. "**Use Case Points Calculator – PETALO".** Disponível em <http://sourceforge.net/projects/uuiet>. Acesso em 14 de março de 2006.

(**Karlsoon, 1997**) Karlsoon J., Ryan Kevin. "**A Cost-Value Approach for Prioritizing Requirements**". In IEEE Software. September/October 1997.

(**Karner, 1993**) Karner G. "**Use Case Points – Resource Estimation for Objectory Projects**". In: Objective System SF AB, 1993.

(**Kiczles, 1997**) Kiczles G., Lamping J., Mendhekar A., Maeda C., Lopes C., Loingtier J., Irwin J. "**Aspect-Oriented Programming**". In European Conference on Object-Oriented Programming, Spring Verlag, 1997.

(**Krasna, 1998**) Krasna, M., Rozman I., Stiglic B. "**How to improve the Quality of Software Enginneering Project Management**". In ACM SIGSOFT'98 Software Engineering Notes vol 23. n. 3.

(**Larman, 2007**) Larman, Craig. "**Utilizando UML e Padrões"**. Porto Alegre: Bookman, 3ª edição, 2007.

(**Loconsole, 2004**) Loconsole Annabella. "**Empirical Studies on Requirement Management Measures**". In IEEE – 26th International Conference on Software Engineering – ICSE'04.

(**Martin, 2005**) Martin N., Pearson J., Furumo K. "**IS Project Management: Size, Complexity, Practices and the Project Management Office**". In IEEE Proceeding of the 38th Hawaii International Conference on System Science, 2005.

(**Matson, 1994**) Matson, J., Barret B., Mellichamp J. "**Software Development Cost Estimation Using Function Points**". In IEEE Transactions on Software Engineering, vol. 20, n. 4, April., 1994.

(**Mohagheghi, 2005**) Mohagheghi P., Anda B., Conradi R. "**Effort Estimation of Use Cases for Incremental Large-Scale Software Development**". In ACM ICSE'05, St. Louis, Missouri, USA, 2005.

(**Nelson, 1999**) Nelson M., Clark J., Spurlock M. "**Curing the Software Requirements and Cost Estimating Blues**". In Software Management & Cost Estimating, November – December, 1999.

(**Nuseibeh, 2000**) Nuseibeh B., Easterbrook S. "**Requirements Engineering: A Roadmap**". In ACM Future Engineering. Limerick, Ireland, 2000.

(**Panlilio, 1992**) Panlilio N. "**Software Estimating Using SLIM Tool".** In Proceeding of the 1992 Conference of the Centre for Advanced Studies on Collaborative research, ACM, 1992.

(**PMBOK, 2000**) PMBOK Guide. "**A Guide to the Project Management Body of Knowledge**". In PMI, USA, 2000.

(**Pressman, 2002**) Pressman, R. "**Engenharia de Software**". McGraw-Hill, 5ª edição, 2002.

(**Rashid, 2003**) Rashid A., Moreira A., Araújo J. "**Modularisation and Composition of Aspectual Requirements**". In 2nd International Conference on Aspect-Oriented Software Development, ACM – Pages 11-20, 2003.

(**Rational Rose, 2006)** Rational Software. Disponível em <http://www.rational.com>. Acesso em 21 de outubro de 2006.

(**Rational Unified Process, 2000**) Rational Unified Process, propriedade e direitos reservados da Rational Software Corporation.

(**Saaty, 1991**) Thomas L. "**Método de Análise Hierárquica**". São Paulo: McGraw-Hill, Markon, 1991.

(**Sampaio, 2005**) Sampaio A., Loughran N., Rashid A., Rayson P. "**Mining Aspects in Requirements**". In Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design. Illinois, USA, 2005.

(**Shumate, 1994**) Shumate K., Snyder T. "**Software Project Reporting: Management, Measurement and Process Improvement**". In ACM - Fullerton, California, 1994.

(**Silva, 2005**) Silva L., Leite J. "**Uma Linguagem de Modelagem de Requisitos Orientada a Aspectos**". In WER'2005, Porto, Portugal.

(**Sousa, 2004**) Sousa G. "**Uma Abordagem Direcionada a Casos de Uso para o Desenvolvimento Orientado a Aspectos**". Dissertação – Programa de Pós-Graduação em Ciência da Computação, UFPE, Recife, Pernambuco.

(**StarUML, 2006**) StatUML the OpenSource UML/MDA Plataform. Disponível em <http://staruml.sourceforce.net/en/>. Acesso em 20 de novembro de 2006.

(**Symons, 1991**) Symons, P. R. "**Software Sizing and Estimating MKII FPA (Function Point Analyst)**". John Wiley & Sons, 1991.

(**The Standish Group, 2004**) The Standish Group. – Chaos Knowledge Center. Disponível em <http://www.standishgroup.com/>. Acesso em 22 de abril de 2006.

(**Wasileski, 2005**) Wasileski J. "**Learning Organization Principles & Project Management**". In ACM SIGUCCS'05, Monterey, California.

(**W3C, 2003**) The World Web Consortium. Disponível em <http://www.w3c.org/>. Acesso em 19 de setembro de 2006.