FUNDAÇÃO EDUCACIONAL EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA - UNIFOR

Tatiana Cavalcanti Monteiro

Pontos de Caso de Uso Técnicos (TUCP): Uma Extensão da UCP

Fortaleza

2005

FUNDAÇÃO EDUCACIONAL EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA - UNIFOR

Tatiana Cavalcanti Monteiro

Pontos de Caso de Uso Técnicos (TUCP): Uma Extensão da UCP

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação

Orientador: Arnaldo Dias Belchior

Fortaleza

2005

Resumo da Tese apresentada ao MIA/UNIFOR como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

PONTOS DE CASO DE USO TÉCNICOS (TUCP): UMA EXTENSÃO DA UCP

Tatiana Cavalcanti Monteiro

Agosto / 2005

Orientador: Arnaldo Dias Belchior
Co-orientador: Carlo Giovano Silva Pires
Programa: Ciências da Computação

A engenharia de software recomenda a implantação de atividades de estimativas de tamanho, esforço, prazo e custo, como formas de melhorar o planejamento e o acompanhamento de projetos de software. Apesar de haver várias técnicas de estimativas, a utilização das mesmas em empresas de software ainda não é uma prática tão comum. A técnica UCP (Pontos de Caso de Uso), por exemplo, é aderente a produtos de software orientados a objetos e baseados em casos de uso. No entanto, têm-se encontrado algumas situações, onde há dificuldades de se obter resultados plenamente satisfatórios ao se utilizar a UCP. Este trabalho apresenta uma extensão da técnica UCP – a TUCP (Pontos de Caso de Uso Técnico) – buscando-se um cálculo mais acurado para o esforço de projetos. Além disso, a TUCP permite uma visão mais detalhada de estimativas nas principais etapas do ciclo de vida do software, possibilitando a realização de refinamentos dessas estimativas para um acompanhamento mais efetivo do projeto.

Abstract of Thesis presented to MIA/UNIFOR as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TECHNICAL USE CASE POINTS (TUCP):
THE EXTENSION OF UCP

Tatiana Cavalcanti Monteiro

August / 2005

Advisor: Arnaldo Dias Belchior

Co-Advisor: Carlo Giovano Silva Pires

Department: Computer Science

The software engineering argues the implantation of activities of size estimates, effort, period and cost, as forms of improving the software projects planning and tracking. In spite of there being several techniques of estimates, the use of the same ones in software companies is not still such a common practice. The technique UCP (Use Case Points), for instance, it is adherent to object-oriented software products and based on use cases. However, they have been finding some situations where there are difficulties of obtaining resulted fully satisfactory when using the UCP. This work presents an extension of the technique UCP – that is TUCP (Technical Use Case Points) – being looked for a calculation more refined for the projects effort. Besides, TUCP allows a more detailed vision of estimates in the main stages of the software cycle life, making possible the accomplishment of refinements of those estimates for an accompaniment more effective of the project.

MONTEIRO, TATIANA CAVALCANTI

Pontos de Caso de Uso Técnicos (TUCP): uma Extensão da UCP
[Fortaleza] 2005

xv, 198p. 29,7 cm (MIA/UNIFOR,
M.Sc., Engenharia de Software, 2005)

Dissertação - Universidade de Fortaleza, MIA

1. Casos de Uso
2. CMMI
3. Estimativas de Software
4. Pontos de Caso de Uso (UCP)
5. Pontos de Caso de Uso Técnicos (TUCP)
6. Progresso Funcional de Software

I. MIA/UNIFOR II. TÍTULO (série)

Dedico este trabalho aos meus queridos pais que sempre me apoiaram e deram provas de carinho, respeito e dedicação.

AGRADECIMENTOS

A Deus, pelos desafios que tem colocado diante de mim.

Ao professor Arnaldo Dias Belchior, pela sua inestimável orientação, estando sempre disponível e efetivamente comprometido com a qualidade deste trabalho.

Ao Carlo Giovano Silva Pires, pela sua co-orientação, estando disponível e efetivamente comprometido com a qualidade deste trabalho.

Aos professores Geraldo Xexéo e Plácido Rogério pela presença na banca examinadora.

Aos meus pais Paulo Leopoldo e Jaidete, também pelos incentivos e conselhos nos momentos difíceis.

A minha tia Maria Éster Monteiro pela dedicação e disponibilidade nas correções finais deste trabalho.

Ao meu noivo Vladmir Pinheiro pelos incentivos e por compreender os momentos de ausência, demonstrando companheirismo, paciência e carinho.

Aos amigos do mestrado, pelas ajudas nos momentos mais difíceis e pelo aprendizado que me proporcionaram.

Aos demais professores do mestrado, pela constante presença e contribuições indiretas.

À secretaria do MIA, pela atenção e presteza sempre imediatas.

Aos demais amigos aqui não citados, mas que certamente foram importantíssimos para a conclusão deste trabalho.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

Este capítulo apresenta as principais questões que motivaram a realização do presente trabalho, como também os objetivos a alcançar e sua organização.

1.1 Motivação

Os problemas relativos à previsibilidade de projetos constituem-se uma das maiores preocupações da indústria de software. O *Standish Group* (2005) relatou que, em 1994, apenas 16% dos projetos de software foram bem sucedidos, ou seja, atenderam a seu objetivo dentro do cronograma e do orçamento previstos. Os dados de 2001 mostram as seguintes estatísticas: 27% de projetos foram finalizados no tempo e custo previstos; 40% de projetos foram cancelados antes de serem finalizados; 50% dos projetos custaram em média 180% a mais da estimativa original. Ocorreu uma melhoria nas estatísticas de projetos bem sucedidos de 1994 (16%) para 2001 (27%). Essa melhoria ocorreu devido aos investimentos da indústria de software na implantação de processos e de qualificação em modelos da qualidade de software.

Embora tenham ocorrido melhorias, a indústria de software continua lidando com projetos mal sucedidos. O CHAOS *Report* de 2003 (STANDISH GROUP, 2005) apresentou os seguintes dados: apenas 34% dos projetos foram bem sucedidos; 15% dos projetos foram cancelados; o erro médio foi de 43% em relação ao orçamento dos projetos concluídos; e apenas 52% das características (requisitos funcionais e não-funcionais) acordadas foram entregues no produto.

Por outro lado, Glass (2003) afirma que algumas das principais causas dos problemas citados são estimativas baseadas em informações insuficientes (usualmente sem que a especificação de requisitos esteja disponível), projetos com requisitos demasiadamente flutuantes e técnicas de estimativas inadequadas.

Um dos principais riscos que atingem os processos de estimativas é a falta de credibilidade em suas estimativas pelas equipes de desenvolvimento (BOEHM, 2000). Isto ocorre quando as estimativas são irreais, ou seja, quando os projetos são subestimados ou superestimados. Neste contexto, as estimativas de tamanho são de fundamental importância para a elaboração de um

cronograma e orçamento realistas, pois as estimativas de tamanho constituem a base para a derivação das estimativas de custo e de esforço (SEI, 2002).

Para garantir o sucesso com o cumprimento dos prazos e custos aceitáveis, faz-se necessário um planejamento bem elaborado do projeto, de forma a medir o tempo real, que será necessário para realizar cada tarefa, garantindo a qualidade dos artefatos gerados ao longo do projeto.

Um dos maiores problemas encontrados no processo de desenvolvimento de software sempre foi e continua sendo a determinação correta do tempo e do esforço necessária para um projeto. Para tanto, saber o tamanho do software é o pré-requisito básico. Considerando-se o momento em que há apenas uma proposta para sua construção, o problema maior é estimá-lo.

Pesquisas vêm sendo desenvolvidas para se determinar formas apropriadas de medir esta grandeza, de maneira proporcional a sua importância. Cabe mencionar, entretanto, que não é possível medir ou estimar características de software, sem que exista um método de desenvolvimento definido, em cada uma de suas fases, entradas, atividades e saídas, pois são eles que estabelecem os pontos de controle para que os dados necessários possam ser obtidos ao longo do processo. É importante observar que estimativas dependem também de informações de projetos anteriores – que devem ser gerenciadas e constantemente atualizadas – em padrões comparáveis quanto à metodologia empregada e ao desempenho de desenvolvedores individuais, o que mais uma vez deixa evidente a necessidade de uma forma definida de trabalho.

Várias métricas de tamanho de software têm sido desenvolvidas e melhoradas desde o final da década de 1960. Com a popularização da orientação a objetos, os conceitos sobre as estimativas efetivamente foram sofrendo alterações para acompanhar as novas mudanças na maneira de desenvolver software. Assim, a técnica UCP (*Use Case Points*) que surgiu nos anos noventa, tem sido aprimorada e vem apresentando crescente utilização. Essa técnica é adequada a projetos, que descrevem seus requisitos de software através de casos de uso.

A UCP tem como proposta ser utilizada logo no início do ciclo de desenvolvimento, na fase de definição dos requisitos, com base no modelo de casos de uso. O modelo de casos de uso é uma técnica largamente utilizada na indústria para capturar e descrever os requisitos funcionais de um software. Esse modelo consiste de diagramas e descrições de casos de uso (ANDA, 2002).

A modelagem dos casos de uso, comumente, é uma das primeiras etapas nos sistemas OO (orientado a objeto), donde advém a praticidade do método UCP, que é permitir a possibilidade de estimar o tamanho do sistema ainda na fase inicial da análise. Fica claro então que quanto mais bem definidos os casos de uso, mais precisa será a estimativa. Entretanto, o método é dependente de casos de uso bem escritos e bem estruturados, com um nível de detalhamento apropriado. Por esta

razão, a inexistência de padrões para a construção de casos de uso pode vir a impactar na aplicação do método.

Este trabalho apresenta uma extensão da técnica UCP, a TUPC (*Technical Use Case Point*), propondo um cálculo mais acurado para o cálculo do esforço de projetos, e permitindo uma visão mais detalhada das estimativas por etapa do ciclo de vida do software, possibilitando a realização de refinamentos dessas estimativas para um acompanhamento mais efetivo do projeto. Além disso, a TUCP propõe calibrações nos fatores de produtividade nessas etapas, permitindo assim a obtenção de estimativas mais acuradas.

1.2 Objetivos

O *objetivo geral* deste trabalho é apresentar a TUCP, buscando-se um cálculo mais acurado para o esforço de projetos, e permitindo uma visão mais detalhada das estimativas por etapa do ciclo de vida do software.

Os objetivos específicos deste trabalho são:

- propor um guia para a especificação de casos de uso, pois isto influencia fortemente o cálculo da estimativa de tamanho;
- refinar a contagem de pontos por casos de uso complexos, para evitar valores subestimados, quando o número de transações é muito grande;
- desatrelar os fatores de ambiente do cálculo do tamanho, por entender que o tamanho é uma grandeza física e não deveria ter seu valor alterado em função desses fatores;
- granularizar o cálculo do tamanho do projeto por cada etapa do processo de desenvolvimento.

1.3 Metodologia

Para a realização deste trabalho, foi feito um estudo bibliográfico na literatura, visando o entendimento teórico necessário sobre métricas de estimativa de tamanho de software, envolvendo principalmente a UCP.

Foram analisados dezenas de projetos de software, que utilizaram a técnica UCP, em uma organização de software certificada CMM, nível 2. As investigações se concentraram nas principais dificuldades e problemas de aplicação da UCP, e de como colher resultados satisfatórios com essa técnica. Uma das conclusões deste estudo foi a constatação de que projetos que continham casos de uso complexos estavam sendo em geral subestimados – já se tinha tido conhecimento disto em outras organizações.

A partir daí, foi proposta inicialmente uma nova maneira de estimar o tamanho de casos de uso que eram considerados muito complexos. Essa proposta foi refinada ao longo dos experimentos realizados e acrescentados outros itens considerados importantes.

1.4 Organização do trabalho

Este trabalho está organizado como se segue.

No Capítulo 2, **Casos de Uso**, encontram-se conceitos sobre modelagem de requisitos através de casos de uso, segundo os padrões da OMG e do contexto da engenharia de software, destacando as melhores práticas para a descrição dos requisitos funcionais de sistemas.

No Capítulo 3, **O CMMI**, apresentam-se conceitos e a estruturação deste modelo de maturidade e capacidade do software, além de apresentar as áreas de processo desse modelo que tratam as questões de estimativas para projetos de software.

No Capítulo 4, **Estimativas de Projeto de Software**, discorre-se sobre algumas importantes técnicas de estimativas para projetos de software, incluindo a técnica UCP, que embasa o modelo proposto.

No Capítulo 5, **TUCP – uma Extensão da UCP**, descreve-se a técnica TUCP e se apresenta uma maneira de fazer acompanhamento de estimativas baseado nessa técnica.

No Capítulo 6, **Estudo de Casos**, estão os principais resultados obtidos pela aplicação da técnica TUCP em cinco projetos de software reais.

No Capítulo 7, **Conclusão**, encontram-se as principais conclusões deste trabalho e perspectivas futuras.

No Apêndice A, é apresentado um modelo para especificação de caso de uso.

No Apêndice B, está um guia contendo orientações de como preencher o modelo de especificação de caso de uso.

Nos Apêndices C, D e E apresenta-se a descrição de três exemplos de casos de uso, utilizando o modelo proposto do Apêndice A.

No Apêndice F, é apresentado um modelo para descrever regras de negócio referentes a um projeto de software.

Capítulo 2

Casos de Uso

Este capítulo apresenta os principais conceitos sobre casos de uso, destacando algumas das melhores práticas de como escrevê-los e propondo um modelo para a especificação dos mesmos.

A modelagem de casos de uso tem sido em geral uma das atividades da fase inicial do processo de desenvolvimento de software (orientado a objetos), realizada para representar requisitos funcionais de um sistema. Esta representação compõe-se de atores, casos de uso e dos relacionamentos entre eles. Os atores desempenham um papel no sistema ou lhe são uma entidade externa. Os casos de uso representam as funcionalidades de um sistema (OMG, 2003).

Os casos de uso são utilizados para descrever requisitos visíveis de um sistema, na fase de análise de requisitos de um projeto e contribuem para o plano de teste e manuais de usuários. Podem auxiliar na elaboração e validação de uma proposta de projeto, bem como na garantia de que todos os requisitos levantados tenham sido tratados. Os casos de uso também são utilizados para a elaboração do cronograma do projeto, cujo plano de projeto descreverá o que será disponibilizado para o cliente (SCHNEIDER *et al.*, 2001). Podem ser considerados como a base para todas as atividades do ciclo de vida do projeto.

A seguir, serão apresentados os principais conceitos sobre casos de uso.

2.1 Ator

Um ator representa um papel que uma pessoa, um dispositivo de hardware ou um outro sistema pode desempenhar em relação a um sistema. Os nomes dos atores devem exprimir claramente o papel que desempenham (RUP, 2003) e são representados por um *stickman* (Ver Figura 2.2).

Para se identificar os atores de um sistema, deve-se pensar primeiramente em seus possíveis usuários, além de como esses atores podem ser categorizados. Normalmente, é tida como uma boa prática pensar em alguns usuários (dois ou três) e verificar se os atores

identificados vão cobrir as necessidades desses usuários. As seguintes questões podem ajudar na identificação dos atores de um sistema (RUP, 2001):

- *Quem vai fornecer, usar ou remover informações*?
- *Quem usará uma determinada funcionalidade*?
- *Quem está interessado em um certo requisito*?
- *Em que parte da organização o sistema é usado*?
- *Quem vai dar suporte e manter o sistema*?
- *Quais são os recursos externos do sistema*?
- *Que outros sistemas necessitarão interagir com este*?

Localizar os atores também auxilia no estabelecimento das fronteiras, na compreensão da finalidade e da extensão do sistema. Vale ressaltar que apenas aqueles que se comunicam diretamente com o sistema precisam ser considerados como atores. Se forem incluídos mais papéis que os do ambiente do sistema, pode-se estar tentando modelar o negócio no qual o sistema será usado, e não apenas o próprio sistema (RUP, 2001).

A localização dos atores é um dos primeiros passos da definição do uso do sistema e cada tipo de fenômeno externo com o qual o sistema deverá interagir poderá vir a ser representado por um ator.

No princípio, talvez seja mais difícil a identificação dos atores mais adequados. Além disso, é provável que não se identifiquem todos de imediato, porque ainda não se têm todos os casos de uso. A partir dos casos de uso identificados, passa-se a ter uma compreensão mais amiúde do ambiente do sistema. Neste ínterim, convém revisar o modelo original, pois há uma tendência natural para se modelar um número excessivo de atores.

Deve-se ter cuidado ao se alterar os atores de um sistema, pois estas alterações efetuadas também podem afetar os casos de uso. Qualquer modificação nos atores pode-se constituir em uma modificação importante nas interfaces e no comportamento do sistema.

É recomendável a nomeação e a descrição dos atores identificados – o nome do ator deverá indicar claramente o seu papel. Para a definição de cada ator, será fornecida uma descrição resumida que inclua a área de responsabilidade e o uso que o ator fará do sistema. Como os atores representam elementos externos ao sistema, não é necessário descrevê-los de forma detalhada (RUP, 2001).

No metamodelo definido pela OMG (2003), um ator possui um nome, pode comunicar-se com um conjunto de casos de uso e ter um conjunto de interfaces, que

descrevem como outros elementos podem comunicar-se com ele. Pode possuir também relacionamentos de generalização com outros atores. Veja Figura 2.1.

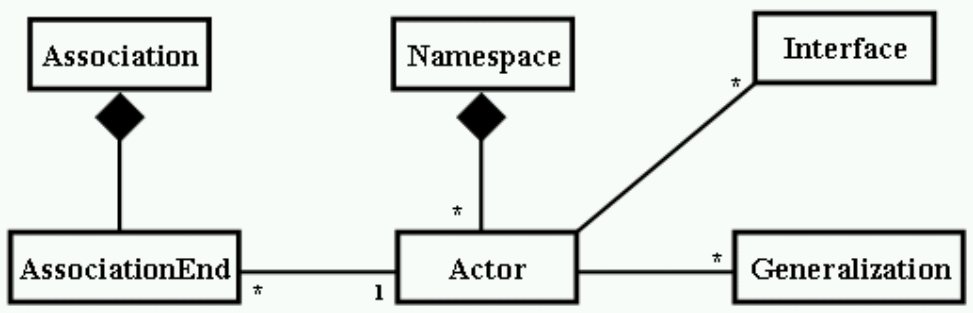


Figura 2.1: Representação do ator no metamodelo da OMG (2003)

Sempre que um usuário específico interage com uma entidade, um certo papel está sendo representado. Uma instância de um ator é um usuário específico interagindo com a entidade. Se a entidade é um sistema, os atores podem representar os usuários humanos ou outros sistemas (OMG, 2003).

Segundo a OMG (2003), instâncias de atores comunicam-se com a entidade através de envio e recebimento de instâncias de mensagem e de instância de caso de uso. Isto é expresso através de associações entre o ator e o casos de uso. Além disso, interfaces podem ser conectadas a um ator, através da definição de como outros elementos podem interagir com o ator. Veja Figura 2.1.

Dois ou mais atores podem comunicar-se com o mesmo conjunto de casos de uso. O compartilhamento de características comuns pode ser expresso através de generalizações de outro ator (possivelmente abstrato), onde são modelados os papéis comuns.

Daí a importância de serem apresentados os relacionamentos entre atores e casos de uso (associação), e os relacionamentos entre atores (generalização).

2.1.1 Associação

Os casos de uso e os atores interagem por meio de sinais enviados entre si. Para indicar tais interações, tem-se a associação de comunicação (ou simplesmente associação) entre o caso de uso e o ator. Um caso de uso tem no máximo uma associação com um determinado ator e vice-versa, seja qual for o número de transmissões de sinais. A rede completa dessas associações é uma imagem estática da comunicação entre o sistema e seu ambiente (RUP, 2001).

As associações não recebem nomes. Como só pode haver uma associação entre um caso de uso e um ator, pode-se especificar o ponto inicial e o ponto final para caracterizar uma

determinada associação. Cada extremidade de uma associação especifica a função que o caso de uso ou o ator desempenha na associação (JACOBSEN, 1992).

Uma associação entre um ator e um caso de uso é exibida por uma linha (ou uma seta) sólida entre o ator e o caso de uso (Figura 2.2). A ponta da seta indica quem inicia cada interação. A utilização de apenas uma linha sólida (sem pontas de setas) poderá indicar também que ator e caso de uso podem iniciar cada interação.

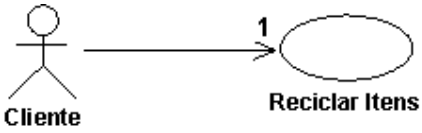


Figura 2.2: Comunicação entre ator e caso de uso

Depois de iniciado, o caso de uso pode comunicar-se com vários atores. É permitido utilizar associações entre o caso de uso e os atores para mostrar com que atores o caso de uso se comunica. A multiplicidade da associação mostra o número de instâncias de um ator com as quais uma instância de caso de uso pode comunicar-se ao mesmo tempo.

2.1.2 Generalização

A generalização de um tipo de ator (descendente) para outro ator (ascendente) indica que o descendente herda o papel que o ascendente pode desempenhar em um caso de uso (RIBU, 2001). Uma generalização entre atores é exibida por uma seta de generalização, isto é, uma seta com a cabeça oca e fechada, que aponta para o ator mais geral.

Na Figura 2.3, os atores "Caixa" e "Contador" são especializações do ator "Supervisor de Saldo". Ambos herdam todas as propriedades do ator pai, podendo atuar como "Supervisor de Saldo".

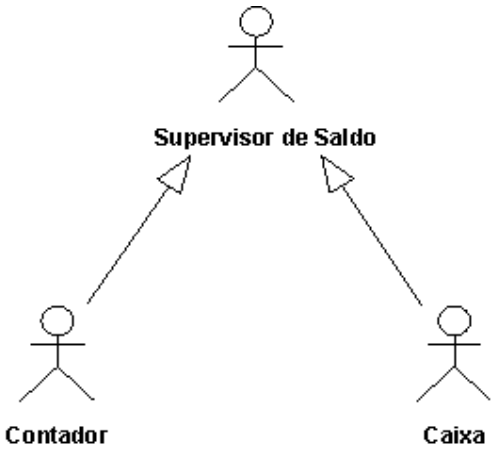


Figura 2.3: Generalização de atores

A seguir, serão apresentados enfoques sobre os casos de uso.

2.2 Enfoques sobre Casos de Uso

Um caso de uso define uma seqüência de ações executadas por um sistema que produz um resultado observável de valor para um ator. Cada caso de uso deve (RUP, 2003):

- *descrever ações que agregam valor para um ator*;
- *mostrar as funcionalidades do sistema a serem usadas por um ator*;
- *modelar um diálogo ente sistema e ator*;
- *possuir fluxo de eventos completo e inteligível da perspectiva de um ator em particular*.

Segundo a OMG (2003), o propósito de um caso de uso é definir parte do comportamento de uma entidade sem revelar a estrutura interna da mesma. Uma entidade especificada deste modo pode ser um sistema ou qualquer elemento que contém comportamento como, por exemplo, um subsistema ou uma classe.

Um caso de uso descreve as interações entre os usuários e a entidade como também as respostas de execução pela entidade, além de como essas respostas são percebidas de fora da entidade. Um caso de uso também inclui possíveis variações em seu fluxo.

Uma instância de caso de uso é uma execução de um caso de uso, iniciado pela instância de uma mensagem de uma instância de ator. Em resposta, a instância do caso de uso executa uma seqüência de ações, conforme especificado pelo caso de uso, através da comunicação das instâncias de ator (OMG, 2003).

As instâncias de atores podem enviar novas instâncias de mensagem para a instância do caso de uso e a interação continuar até a instância ter respondido a todas as entradas.

Segundo a OMG (2003), um caso de uso pode ser representado com compartimentos, exibindo seus atributos e operações, conforme a Figura 2.4.

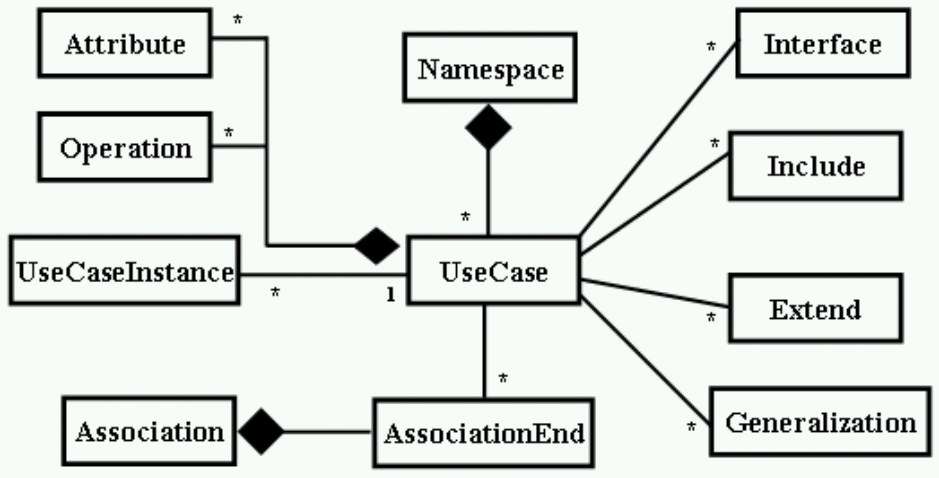


Figura 2.4: Representação do caso de uso no metamodelo da OMG (2003)

Um caso de uso pode ser representado através de uma elipse contendo o nome do caso de uso. Uma representação adicional é colocar o nome do caso de uso abaixo da elipse (Figura 2.5). Essa representação pode conter ou pode suprimir os compartimentos que apresentam os atributos, as operações e os pontos de extensão do caso de uso.

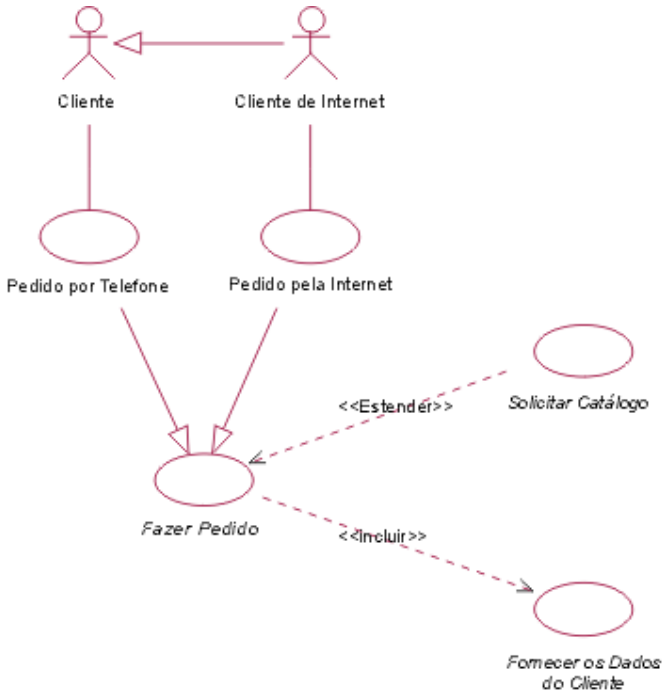


Figura 2.5: Modelo de caso de uso

O comportamento de um caso de uso pode ser descrito de diferentes modos, dependendo da conveniência. A maneira mais comum de descrever um caso de uso é de forma textual. As máquinas de estado, operações e métodos são exemplos de outros modos de

se descrever o comportamento do caso de uso. Já os diagramas de seqüência podem ser usados para descrever a interação entre casos de uso e seus atores (KRUCHTEN, 2001).

A melhor forma de se encontrar um modelo "ideal" de caso de uso é desenvolver dois ou três modelos, escolher o mais apropriado e aperfeiçoá-lo. O desenvolvimento de modelos alternativos também auxilia a compreender melhor o sistema. Após descrever o primeiro modelo de casos de uso, deve-se verificar se ele abrange todos os requisitos funcionais que foram levantados a partir das características do projeto. Os requisitos devem ser cuidadosamente examinados, para assegurar que todos os casos de uso satisfaçam a todos os requisitos, e que os requisitos, por sua vez, atendam a todas as características observadas no projeto.

Cada caso de uso deve ter um nome que o identifique e indique o que é alcançado em suas interações com os atores. O nome do caso de uso pode necessitar de um conjunto de palavras para ser entendido. Todavia, deve-se atentar para a concisão desse nome e para que dois casos de uso não sejam nomeados da mesma forma.

Um caso de uso é um comportamento do sistema, que produz um resultado mensurável de valor para um ator. Os casos de uso descrevem aquilo que os atores querem que o sistema execute para eles. Deve ser uma tarefa completa na perspectiva de um ator. Portanto, cada caso de uso identificado deverá estar associado a pelo menos um ator (SCHNEIDER *et al*., 2001).

É importante não confundir os casos de uso com a chamada decomposição funcional, a qual constitui-se de uma boa maneira de particionamento de um problema em partes menores. Essas partes em conjunto fornecem a funcionalidade do sistema. Já os casos de uso fornecem uma narrativa de como os atores utilizam o sistema. Embora essas duas técnicas descrevam requisitos, seus propósitos diferem entre si.

No início de sua elaboração, os casos de uso ainda são incipientes; certamente, eles precisarão ser alterados algumas vezes até que sejam estabilizados. Se a visão ou os requisitos do sistema forem deficientes, ou ainda, se a análise do sistema for vaga, seguramente as funcionalidades pretendidas para o sistema serão confusas. Assim sendo, os analistas de sistemas deverão estar preparados para adicionar, remover, reunir e dividir casos de uso antes de chegar a uma versão final (estabilizada) dos mesmos. Após descrevê-los com detalhes, os casos de uso poderão ser mais bem compreendidos. A seguir, serão apresentados os tipos de relacionamentos entre casos de uso.

2.3 Relacionamentos entre Casos de Uso

Um caso de uso pode apresentar três tipos de relacionamentos: inclusão, extensão, e generalização. Nesses relacionamentos, o caso de uso original, que é modificado, é chamado de caso de uso base.

Outro conceito apresentado pelo RUP (2001) é que um caso de uso pode ser dito concreto e abstrato. Um caso de uso concreto é iniciado por um ator e constitui um fluxo completo de eventos. "Completo" significa que uma instância do caso de uso executa a operação inteira chamada pelo ator.

Já um caso de uso abstrato propriamente nunca é instanciado. Os casos de uso abstratos são incluídos em (Ver seção 2.3.1: Inclusão), se estendem para (Ver seção 2.3.2: Extensão) ou generalizam (Ver seção 2.3.3: Generalização) outros casos de uso. Quando um caso de uso concreto é iniciado, uma instância do caso de uso é criada. Essa instância também exibe o comportamento especificado por seus casos de uso abstratos associados. Portanto, nenhuma instância separada é criada de casos de uso abstratos (RUP, 2001).

A distinção entre os dois é importante porque são os casos de uso concretos que os atores "verão" e iniciarão no sistema.

Todos os casos de uso abstratos são do tipo inclusão, extensão, ou generalização, não sendo necessário ter um ator associado a esses casos de uso.

Uma associação de comunicação com um ator só se faz necessária, se o comportamento dos casos de uso envolver de forma explícita a interação com esse ator. Neste contexto, estes relacionamentos se tornam casos de uso concretos.

2.3.1 Inclusão

O relacionamento de inclusão (*include*) conecta um *caso de uso base* a um *caso de uso de inclusão*. Um caso de uso de inclusão descreve um segmento de comportamento que é inserido em uma instância de caso de uso ao ser executado o caso de uso base, o qual insere explicitamente o caso de uso de inclusão, mas nenhum deles pode acessar os atributos um do outro (KRUCHTEN, 2001).

O comportamento da inclusão é inserido em um local específico no caso de uso base. Quando uma instância do caso de uso chegar ao local no caso de uso base, onde o relacionamento de inclusão é definido, ela passará a seguir a descrição do caso de uso de inclusão. Uma vez realizada a inclusão, a instância de casos de uso retomará as atividades do ponto em que parou no caso de uso base.

O relacionamento de inclusão não é condicional. Se a instância do caso de uso chegar ao local no caso de uso base, onde ele é definido, ela será sempre executada. Caso se deseje expressar uma condição em um caso de uso, essa condição poderá ser colocada em fluxo alternativo no caso de uso base. Se a instância do caso de uso nunca chegar ao local onde o relacionamento de inclusão é definido, esta não será executada (SCHNEIDER *et al*., 2001).

Em várias situações, para compartilhar o mesmo comportamento, que é comum a dois ou mais casos de uso, pode-se fatorar esse comportamento em um caso de uso de inclusão. Neste sentido, quando se tiver um conjunto de passos comuns a vários casos de uso, esse conjunto pode ser um bom candidato a um caso de uso de inclusão. Portanto, um caso de uso de inclusão pode ser inserido em diversos casos de uso base, embora isso não indica nenhum relacionamento entre esses casos de uso base.

Pode até haver diversos relacionamentos de inclusão entre o mesmo caso de uso de inclusão e um mesmo caso de uso base, desde que a inclusão seja feita em diferentes locais no caso de uso base. Esses casos de uso de inclusão podem ser aninhados, podendo um caso de uso de inclusão funcionar como caso de uso base para um outro caso de uso de inclusão (KRUCHTEN, 2001).

Se parte de um caso de uso base representar uma função pela qual o caso de uso somente dependa do resultado e não do método usado para produzir esse resultado, pode-se transformar essa parte em um caso de inclusão.

Quando um fluxo de eventos é longo (tipicamente maior que uma página), pode-se tentar reduzi-lo, com o propósito de melhorar seu entendimento. Isto pode ser feito através da identificação de um subfluxo desse fluxo, que tenha um conjunto de passos coerente com um alvo, podendo também ser encapsulado em caso de uso de inclusão.

O relacionamento de inclusão pode ser utilizado ainda para separar o comportamento de um caso de uso base, que não seja necessário para a compreensão de sua finalidade principal – apenas o resultado é importante.

Um passo mais complexo poderia ser também um candidato a um caso de uso de inclusão, para evitar o "congestionamento" do cenário principal. No entanto, devem-se evitar muitas subdivisões de casos de usos, que possam vir a caracterizar uma decomposição funcional.

Uma relação de inclusão entre casos de uso é apresentada por uma seta tracejada de ponta aberta do caso de uso base para o caso de uso inclusão. A seta é marcada com a palavra chave «incluir»(«*include*»), conforme a Figura 2.6.

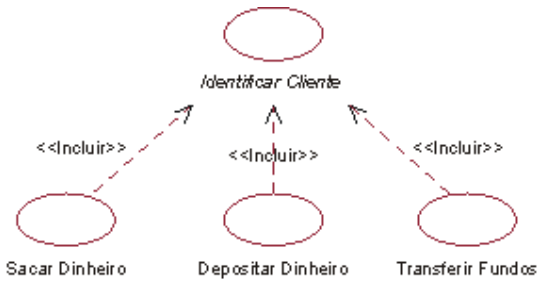


Figura 2.6: Relacionamento de inclusão

2.3.2 Extensão

O relacionamento de extensão estabelece a conexão entre um *caso de uso de extensão* e um *caso de uso base* (KRUCHTEN, 2001).

Um relacionamento de extensão é aquele que se estabelece entre um caso de uso de extensão e um caso de uso base, especificando como o comportamento definido para o caso de uso de extensão pode ser inserido no comportamento definido para o caso de uso de base. Ele é inserido implicitamente no sentido de que a extensão não é exibida no caso de uso base (RUP, 2001).

A extensão deve ser inserida fazendo referência aos pontos de extensão (seção 2.7) no caso de uso base (RUP, 2003).

As extensões podem ser usadas para diversas finalidades (RUP, 2003):

- mostrar que parte de um caso de uso é um comportamento opcional (ou possivelmente opcional) do sistema. Isso faz a diferenciação entre comportamento opcional e comportamento obrigatório em um modelo;
- indicar que um subfluxo só é executado em determinadas condições (algumas vezes excepcionais), como o disparo de um alarme;
- apontar que pode haver um conjunto de segmentos de comportamento dentre os quais um ou vários podem ser inseridos em um ponto de extensão de um caso de uso base. Os segmentos de comportamento que são inseridos (e a ordem pela qual são inseridos) dependerão da interação com os atores durante a execução do caso de uso base.

A extensão é condicional, o que significa que a execução depende do que tiver acontecido durante a execução do caso de uso base. O caso de uso base não controla as condições da execução da extensão; essas condições são descritas no relacionamento de extensão. O caso de uso de extensão pode acessar e modificar atributos de seu caso de uso

base. O caso de uso base, porém, não pode ver as extensões nem acessar seus atributos (SCHNEIDER *et al*., 2001).

O caso de uso base é modificado implicitamente pelas extensões. Também, pode-se dizer que o caso de uso base define um *framework* modular ao qual podem ser adicionadas extensões; todavia, essas extensões específicas não ficam visíveis no caso de uso base (RUP, 2001).

O caso de uso base deve ser completo em si mesmo, o que significa que deve ser compreensível e fazer sentido sem nenhuma referência a extensões. No entanto, ele não é independente das extensões, já que não pode ser executado sem que as extensões também possam ser executadas (RUP, 2001).

O caso de uso de extensão pode consistir em um ou mais segmentos de inserção, sendo que cada um deles pode ter caminhos alternativos internos. Esses segmentos modificam gradativamente o comportamento do caso de uso base. Cada segmento de inserção é um caso de uso de extensão, que pode ser inserido em um local separado no caso de uso base. Isso significa que o relacionamento de extensão tem uma lista de referências a pontos de extensão, igual em termos de quantidade ao número de segmentos de inserção do caso de uso de extensão. Cada ponto de extensão deve ser definido no caso de uso base (OMG, 2003).

Um caso de uso base pode estar ligado a vários relacionamentos de extensão, o que significa dizer que uma instância de caso de uso pode executar mais de uma extensão durante sua vida útil. Um caso de uso de extensão pode relacionar-se com vários casos de uso base; o que não implica, entretanto, na existência de qualquer dependência entre esses casos de uso base.

Um caso de uso de extensão pode ser, por si só, a base de um relacionamento de extensão, de inclusão ou de generalização. Isto significa, por exemplo, que os casos de uso de extensão podem desdobrar-se em outros casos de uso de extensão de maneira aninhada (COCKBURN, 2001).

O caso de uso de extensão, assim como qualquer caso de uso, pode ter um fluxo básico e fluxos alternativos. O caminho exato que a instância de caso de uso percorrerá na extensão dependerá não somente do que tiver acontecido antes da execução (o estado da instância de caso de uso), mas também do que acontecerá na interação com os atores à medida que a extensão for executada. Depois que a instância de caso de uso tiver executado a extensão, ela volta a executar o caso de uso base, no ponto em que o havia deixado (JACONSON *et al*., 2000).

Um caso de uso de extensão pode ter mais de um segmento de inserção, cada um deles relacionado a seu próprio ponto de extensão no caso de uso base. Se for este o caso, a instância de caso de uso retoma o caso de uso base e continua a executá-lo até o próximo ponto de extensão especificado no relacionamento de extensão. Neste ponto, será executado o próximo segmento de inserção do caso de uso de extensão. Este processo se repete até que o último segmento tenha sido executado. É importante observar que a condição do relacionamento de extensão é verificada apenas no primeiro ponto de extensão. Se ela for verdadeira ou ausente, a instância de caso de uso deve executar todos os segmentos de inserção (JACONSON *et al*., 2000). Se a condição do relacionamento de extensão for falsa, ele não é executado (RUP, 2003).

Portanto, pode haver até vários relacionamentos de extensão entre o mesmo caso de uso de extensão e o caso de uso base, que a extensão seja inserida em diferentes locais do caso de uso base. Para isto é necessário que os diferentes relacionamentos de extensão façam referência a diferentes pontos de extensão, no caso de uso base.

Uma relação de extensão entre casos de uso é exibida através de uma seta tracejada de ponta aberta, do caso de uso provendo a extensão para o caso de uso base. A seta é marcada com a palavra chave «estender» («*extend*»), conforme Figura 2.7. A condição da relação pode ser opcionalmente apresentada perto da palavra chave.

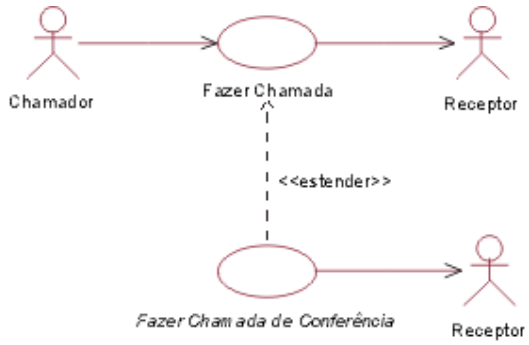


Figura 2.7 - Relacionamento de extensão

2.3.3 Generalização

Uma generalização de casos de uso é um relacionamento de um caso de uso filho com um caso de uso pai, especificando como um filho pode adotar todo o comportamento e as características descritas pelo pai (OMG, 2003). Deve-se usar o relacionamento de generalização se os casos de uso pai e filhos compartilharem estruturas e propósitos similares (RUP, 2003).

Um caso de uso pai pode ser especializado em um ou mais casos de uso filhos, que representam formas mais específicas do pai. Nem o pai, nem o filho são necessariamente abstratos, embora o pai seja abstrato na maioria das vezes. Um filho herda a estrutura, o comportamento e os relacionamentos do pai. Além disso, todos os filhos do mesmo pai são especializações do pai (KRUCHTEN, 2001).

A generalização é usada quando há dois ou mais casos de uso que têm comportamento, estrutura e finalidade comuns. Quando isso ocorrer, podem-se descrever as partes compartilhadas em um novo caso de uso, geralmente abstrato, que é especializado pelos casos de uso filhos (KRUCHTEN, 2001).

O caso de uso filho depende da estrutura do caso de uso pai e pode adicionar um comportamento ao pai inserindo os segmentos de comportamento no comportamento herdado ou declarando relacionamentos de inclusão estendidos para o caso de uso filho. O filho pode modificar os segmentos de comportamento herdados do pai, o que precisa ser feito adequadamente, para que a estrutura do caso de uso pai seja preservada. Isto significa dizer que todos os segmentos de comportamento, descritos como passos ou subfluxos do fluxo de eventos do pai, ainda devem existir, mas o conteúdo desses segmentos de comportamento pode ser modificado pelo filho (KRUCHTEN, 2001).

Embora dois casos de uso filho possuam o mesmo pai, suas especializações são independentes umas das outras, o que significa que são executadas em instâncias de casos de uso separadas. Isso é diferente dos relacionamentos de extensão ou de inclusão, em que várias adições modificam implícita ou explicitamente uma instância de casos de uso que executa o mesmo caso de uso base (RUP, 2001).

Tanto a generalização de casos de uso quanto à de inclusão podem reutilizar o comportamento entre os casos de uso no modelo. A diferença é que na generalização de casos de uso, a execução dos filhos depende da estrutura e do comportamento do pai (a parte reutilizada), enquanto que, em um relacionamento de inclusão, a execução do caso de uso base depende apenas do resultado da função que o caso de uso de inclusão (parte reutilizada) executa.

Ainda na generalização, os filhos compartilham similaridades em finalidade e em estrutura, enquanto que no relacionamento de inclusão, os casos de uso base, que estão reutilizando a mesma inclusão, podem ter finalidades completamente diferentes, mas precisam que a mesma função seja executada (RUP, 2003).

A principal diferença entre casos de uso de extensão e generalização é que a extensão modifica o caso de uso (o objetivo pode mudar), enquanto que na generalização o caso de uso permanece o mesmo (o objetivo é preservado entre pai e filho) (RUP, 2003).

Uma instância de caso de uso que executa um caso de uso filho seguirá o fluxo de eventos descritos para o caso de uso pai, inserindo um comportamento adicional e ou modificando o comportamento da maneira definida no fluxo de eventos do caso de uso filho (KRUCHTEN, 2001).

Se um caso de uso é especificado em mais de um caso de uso pai (herança múltipla), deve-se deixar explícito o estado da seqüência de comportamento dos casos de uso nos pais, na especificação do filho. Todavia, a utilização de herança múltipla, embora tecnicamente possível, tem dificultado o entendimento dos requisitos.

O relacionamento de generalização entre casos de uso é mostrado por uma seta de generalização, isto é, uma linha sólida com uma ponta fechada e oca, que aponta para o caso de uso pai, conforme Figura 2.8.

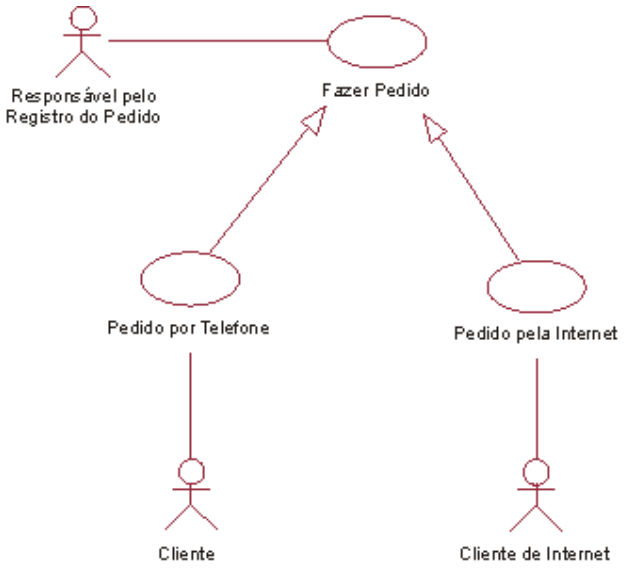


Figura 2.8: Generalização de casos de uso

2.4 Fluxo de Eventos

A melhor forma de identificar casos de uso é considerar o que cada ator exige do sistema uma vez que o sistema existe apenas para seus usuários e deve estar baseado em suas necessidades. Muitas das necessidades dos atores serão reconhecidas através dos requisitos funcionais especificados no sistema. Para cada ator, humano ou não, as seguintes perguntas podem ser consideradas (RUP, 2001):

- *Segundo o ator, quais são as principais tarefas a serem executadas pelo sistema*?
- *O ator criará, armazenará, alterará, removerá ou lerá dados no sistema*?
- *O ator precisará informar o sistema sobre mudanças externas repentinas*?
- *O ator necessitará estar informado sobre certas ocorrências no sistema*?
- *O ator iniciará ou finalizará o sistema*?

As respostas a estas perguntas representam os fluxos de eventos, que identificam sugestões de casos de uso. Nem todas essas respostas constituem casos de uso distintos; algumas podem vir a ser modeladas como variantes de um mesmo caso de uso. Nem sempre é fácil saber o que é uma variante e o que é um caso de uso separado e distinto. (KRUCHTEN, 2001).

O fluxo de eventos poderá documentar a lógica de execução de um caso de uso; é uma forma de detalhar como um caso de uso será iniciado e executado sem ser necessário tornar o diagrama muito extenso e detalhado (OMG, 2003).

Segundo Cockburn (2001), o fluxo de eventos de um caso de uso contém as informações mais importantes derivadas de sua modelagem. Ele deve descrever o passo a passo dos eventos do caso de uso de forma clara, para que alguém externo ao projeto o entenda facilmente.

O fluxo de eventos deve apresentar o que o sistema faz e não como é o *design* do sistema para realizar o comportamento exigido, não devendo conter detalhes de implementação. Quando se fizer necessária uma melhor compreensão, o fluxo de eventos de um caso de uso poderá ser modelado através de um diagrama de atividades

As duas principais partes do fluxo de eventos são o fluxo de eventos básico (ou *fluxo básico*) e os fluxos de eventos alternativos (*fluxos alternativos*). Opcionalmente, podem-se ter também os fluxos de eventos de exceção (*fluxos de exceções*). Esses fluxos podem ser capturados em seções de um documento de especificação de casos de uso (ver sugestão desse tipo de documento no Apêndice A).

O fluxo básico deve abordar o que "geralmente" ocorre quando o caso de uso é executado, enfatizando o cenário de sucesso do início ao fim (cenário *happy day*). É recomendável que o fluxo básico seja relativamente pequeno e de fácil leitura, semelhante a uma pequena história, apresentando os passos necessários para alcançar o objetivo principal do caso de uso (RUP, 2003).

Os itens abaixo poderão auxiliar a tomar decisões sobre o nível de detalhamento de um fluxo básico (RUP, 2003):

- *Que eventos iniciam o caso de uso*?
- *Como o caso de uso finaliza*?
- *Como se dá a repetição de algum comportamento no caso de uso*?

A estrutura do caso de uso é uma boa prática para se elaborar alternativas para o fluxo básico. Para cada passo, pode-se perguntar: *Como isto poderia ser realizado de uma outra forma*? *O que poderia dar errado*? Esses detalhes devem ser colocados em fluxos alternativos e, opcionalmente, em fluxos de exceção.

Os fluxos alternativos abordam o comportamento de caráter opcional ou excepcional em relação ao comportamento normal e também as variações do comportamento normal. Pode-se pensar nos fluxos alternativos como "desvios" ou variantes do fluxo básico, alguns dos quais voltarão ao fluxo básico e alguns finalizarão a execução do caso de uso (KRUCHTEN, 2001; RUP, 2003).

Os itens abaixo também poderão auxiliar a tomar decisões sobre o nível de detalhamento de um fluxo alternativo (RUP, 2003):

- *Há situações opcionais no caso de uso*?
- *Que situações casuais podem surgir*?
- *Que variantes podem acontecer*?
- *O que pode dar errado*?
- *O que não pode acontecer*?
- *Que tipo de recursos pode ser agrupado*?

A Figura 2.9 apresenta a estrutura típica do fluxo de eventos. A seta reta representa o fluxo de eventos básico, enquanto que as setas curvas representam os caminhos alternativos em relação ao normal. Alguns caminhos alternativos voltam ao fluxo de eventos básico, enquanto outros finalizam o caso de uso (RUP, 2001).

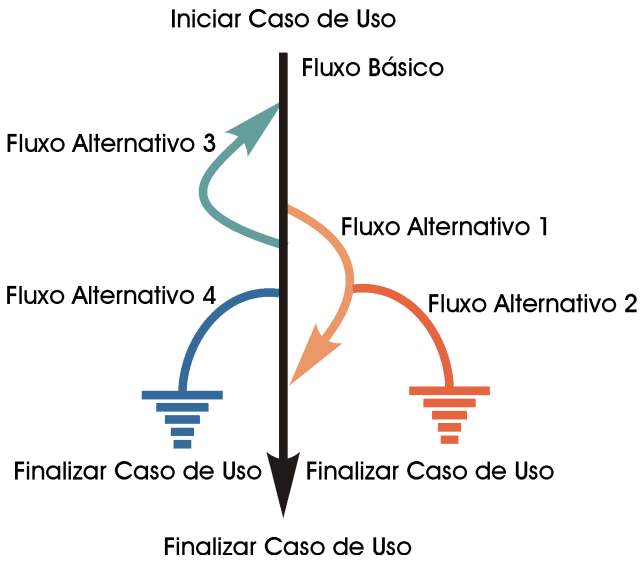


Figura 2.9: Estrutura típica do fluxo de eventos (RUP, 2003)

Tanto o fluxo básico quanto os fluxos alternativos devem ser estruturados em passos e podem ter subfluxos. Com isso, a principal meta deve ser a legibilidade textual do caso de uso. Uma maneira prática de proceder é a seguinte: um subfluxo deve ser um segmento de comportamento no caso de uso, que tem uma finalidade bem definida, e ser "atômico", isto é, nele devem ser executadas todas ou nenhuma das ações descritas. Podem ser necessários vários níveis de subfluxos, mas isso deve ser evitado, sempre que possível, pois torna o texto mais complexo e de difícil entendimento (KRUCHTEN, 2001).

Uma seqüência específica de ações que mapeiam comportamentos em fluxos de eventos pode ser obtida através de cenários. Um cenário é um caminho através do qual fluxos de eventos de caso de uso iniciam em um ponto e terminam em outro. Ele pode envolver o fluxo básico e um número qualquer de combinações de fluxos alternativos.

2.5 Cenário

Um cenário descreve instâncias de uso do sistema (de caso de uso), através de uma seqüência de passos, apresentando as interações entre um ator e o sistema, isto é, uma alternativa do que pode acontecer. Cada passo deve ser uma declaração simples, evidenciando claramente quem o está executando. O passo deve apresentar a intenção do ator e não os mecanismos daquilo que o ator faz. Portanto, a interface com o caso de uso não deve ser descrita (FOWLER, 2005).

Quantos cenários são, então, necessários em um caso de uso? Tantos quantos forem necessários para a compreensão do sistema que está sendo desenvolvido.

Cenários podem ser escritos para entender, bem como validar os fluxos de casos de uso. Muitos, primeiramente, elaboram cenários, extraindo deles os casos de uso. Outros, inicialmente, constroem as especificações de casos de uso e os validam elaborando os cenários. Cenários podem compor excelentes casos de teste. O benefício de se capturar cenários de especificações de casos de uso é que todas as informações requeridas para análise, projeto e testes devem estar descritas no caso de uso. Neste sentido, um documento de especificação de casos de uso poderá ter uma seção com um "conjunto de cenários" (RUP, 2003).

Segundo Bittner e Spence (2002), a utilização de cenários deve:

- ser compatível com os casos de teste;
- corresponder ao que atualmente deve ser executado, isto é, o que o sistema faz na prática;
- ser útil para análise e projeto, auxiliando desenvolvedores a pensar sobre como o sistema será usado.

A documentação dos cenários é útil porque, embora se elaborem fluxos em um caso de uso, implementam-se e testam-se os cenários. Uma grande vantagem do uso de cenários é que um conjunto de cenários poderá vir a resultar em uma família de casos de teste.

Pode-se dar a um cenário um nome descritivo e se fazer a listagem dos fluxos (em ordem) que constituem o cenário. Vale salientar que os cenários não devem conter detalhes de dados, que sejam entradas ou saídas do sistema. Esses dados devem estar presentes nos casos de teste.

A seguir serão apresentadas noções sobre precondições e as pós-condições em casos de uso.

2.6 Precondições e Pós-Condições

Os conceitos de precondição e de pós-condição são utilizados para esclarecer como o fluxo de eventos inicia e termina respectivamente (Figura 2.10). Todavia, esses conceitos somente deverão ser utilizados se vierem agregar algum valor ao público alvo do caso de uso.

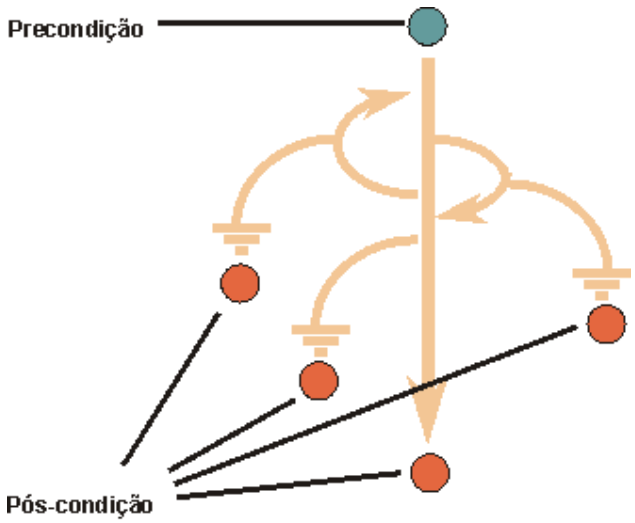


Figura 2.10 - Precondição e pós-condição (RUP, 2003)

A precondição corresponde ao estado do sistema e da sua vizinhança, que é exigido antes do início da execução de um caso de uso. Os estados descritos na precondição devem ser observados pelo usuário. No entanto, nem todos os casos de uso irão precisar de precondição. (KRUCHTEN, 2001).

Uma precondição é uma descrição textual definindo o que o sistema deve garantir como verdadeiro, antes que o caso de uso seja iniciado, ou quando ele for invocado, isto é, que condições necessitam ser verificadas com antecedência.

Uma precondição é também uma restrição sobre quando um caso de uso poderá iniciar ou sobre o evento que o inicia. A precondição não deve ser direcionada apenas para um subfluxo de um caso de uso, apesar de ser possível definir precondições e pós-condições em nível de subfluxo.

A pós-condição é o estado que o sistema pode apresentar após o término do caso de uso. Os estados descritos devem ser observados também pelo usuário (KRUCHTEN, 2001).

Uma pós-condição descreve o que o sistema deverá assegurar ao término do caso de uso. As garantias de sucesso mantêm-se após um cenário bem-sucedido, como também as garantias mínimas devem ser mantidas após qualquer cenário.

A pós-condição de um caso de uso deve ser verdadeira, independentemente dos fluxos alternativos que tenham sido executados, isto é, ela não deve ser verdadeira apenas para o fluxo básico. A possibilidade de falhas deverá ter sido previsto na pós-condição.

Na pós-condição, é necessário atentar para os relacionamentos de extensão, evitando-se que algum subfluxo do caso de uso estendido possa vir a violar a pós-condição do caso de uso base.

2.7 Pontos de Extensão

Um ponto de extensão abre o caso de uso para a possibilidade de uma extensão. Ele tem um nome e uma lista de referências para um ou mais locais no fluxo de eventos do caso de uso. Um ponto de extensão pode fazer referência a um único local entre dois passos do comportamento no caso de uso, como também pode fazer referência a um conjunto de locais diferentes.

Usar pontos de extensão nomeados promove a separação da definição do comportamento do caso de uso estendido dos detalhes internos do caso de uso base. O caso de uso base pode ser modificado ou reorganizado, contanto que os nomes atribuídos aos pontos de extensão permaneçam os mesmos e isso não afete o caso de uso estendido.

Os pontos de extensão simplificam o texto que descreve o fluxo de eventos do caso de uso base com os detalhes do local em que o comportamento pode ser estendido. Os pontos de extensão estão diretamente ligados aos relacionamentos de extensão (seção 2.3.2).

2.8 Requisitos Especiais

Em geral, um requisito especial é um requisito não funcional pertencente a um caso de uso, que não é coberto no texto do fluxo de eventos, por ter havido dificuldades em sua definição. Podem-se citar alguns tipos de requisitos especiais (RUP, 2003):

- requisitos legais e reguladores;
- padrões de aplicativo;
- requisitos de compatibilidade;
- restrições de projeto; e
- atributos de qualidade do sistema.

No RUP, para os atributos de qualidade é utilizada a sigla URPS, que vem do modelo FURPS (*Functionality*, *Usability*, *Reliability*, *Performance*, *Supportability*) (RUP, 2003):

- *Usabilidade*: verifica se o sistema é fácil de ser entendido, aprendido ou operado por diversos tipos de usuários, sob determinadas condições (ISO, 2001). Suas subcaracterísticas são:
  - ✓ fatores humanos;
  - ✓ estética;

- ✓ consistência;
- ✓ documentação do usuário.

- *Confiabilidade*: capacidade de o sistema manter um nível de desempenho satisfatório, sob um certo período de tempo sem apresentar falhas (ISO, 2001). Suas subcaracterísticas são:
  - ✓ freqüência e severidade de falha;
  - ✓ recuperabilidade;
  - ✓ previsibilidade;
  - ✓ acurácia;
  - ✓ tempo médio de falhas (MTBF).
- *Desempenho*: medida da velocidade ou eficiência de operação do sistema (RUP, 2003). Suas subcaracterísticas são:
  - ✓ velocidade;
  - ✓ eficiência;
  - ✓ disponibilidade;
  - ✓ exatidão;
  - ✓ tempo de resposta.
- *Suportabilidade*: habilidade de o sistema ser facilmente modificado para incluir manutenções evolutivas ou corretivas. Suas subcaracterísticas são:
  - ✓ testabilidade;
  - ✓ extensibilidade;
  - ✓ adaptabilidade;
  - ✓ manutenibilidade;
  - ✓ compatibilidade;
  - ✓ configurabilidade;
  - ✓ possibilidade de serviço;
  - ✓ instalabilidade;
  - ✓ localizabilidade;
  - ✓ robustez.

Os atributos que compõem o modelo FURPS fazem parte de características e subcaracterísticas de qualidade de produtos de software, propostas na ISO/IEC 9126 (ISO, 2001).

2.9 Regras de Negócio

As regras de negócio consideradas importantes para o sistema devem ser descritas fora do fluxo de eventos, como um item à parte na Especificação de Casos de Uso (Apêndice A). Caso se justifiquem, seja pela complexidade ou relevância do sistema, seja por orientação da organização, as regras de negócio do sistema poderão ser alocadas em um documento próprio para este fim (Apêndice F).

As regras de negócio podem ser referenciadas nos passos do fluxo de evento do caso de uso. Regras simples e genéricas podem ser descritas nos próprios passos do fluxo de eventos, ou junto aos dados, a exemplo de regras de validação de dados, datas, máscaras, etc.

Uma regra de negócio (RN) poderia ter o seguinte formato:

- *Identificação*: RN_<*seqüencial numérico*>
- *Nome*: denominação que sintetiza a regra de negócio
- *Descrição*: corresponde à regra de negócio propriamente dito

Sejam, por exemplo, as seguintes regras de negócio:

RN_01: *Cálculo de Custos de Projeto*

> Descrição: o custo de um projeto corresponde ao somatório de todos os gastos com recursos humanos alocados a esse projeto no período, adicionando-se os lançamentos de outras despesas a ele referentes. O custo de um recurso é dado multiplicando-se o item *homem.hora* (salário, encargos e custos administrativos) pelo *somatório das horas registradas* para o projeto no período considerado.

RN_02: *Inclusão de membro na equipe de projeto*

> Descrição: um membro somente poderá ser incluído na equipe de um novo projeto, se a soma de suas horas alocadas em outros projetos não ultrapassar 1.800 horas.

A partir das regras de negócio acima, um fluxo de eventos do caso de uso "Gerar planilha de custos do projeto", poderia ter o seguinte passo:

Passo *x*: o sistema processa o *Cálculo de Custos de Projeto* (RN_01).

Já para o caso de uso "Manter equipe", poder-se-ia ter o seguinte passo para o fluxo de eventos:

Passo *y*: o sistema valida a *Inclusão de membro na equipe de projeto* (RN_02).

Após a apresentação dos principais conceitos sobre os elementos que podem compor um caso de uso, descrever-se-ão a seguir algumas orientações sobre especificação de casos de uso.

2.10 Orientações sobre Especificação de Caso de Uso

Embora os casos de uso sejam reconhecidos como parte relevante na UML, esta não descreve como se capturar o conteúdo de um caso de uso, uma vez que praticamente todo o valor dos casos de uso reside em seu conteúdo textual. O diagrama de casos de uso por si só é de valor bastante limitado. Ainda não há uma maneira padronizada para se descrever o conteúdo de um caso de uso. Tem sido recomendado que um caso de uso seja descrito de forma breve e tenha fácil leitura. Um caso de uso rico em detalhes poderia ser justificado em função de seu risco ser elevado (FOWLER, 2005).

A finalidade mais importante de uma especificação de casos de uso é comunicar o comportamento do sistema ao cliente ou ao usuário final. Conseqüentemente, essa especificação deve ser de fácil entendimento e de simples leitura (RUP, 2001).

Um dos pontos críticos e maior dificuldade em uma especificação de caso de uso é o aprendizado de como determinar qual o nível de detalhamento que o caso de uso deva ter: como "começar e terminar"; onde as características do sistema (*features*) terminam e os casos de uso começam; onde os casos de uso terminam e o projeto começa? Os casos de uso como também os requisitos do software devem estabelecer "o que" o sistema executa, mas não "como" ele executa (Figura 2.11).

Os atores e os casos de uso são obtidos a partir dos requisitos elicitados de clientes e de possíveis usuários do sistema. De acordo com esses requisitos, casos de uso e atores devem ser inicialmente descritos. Antes da descrição detalhada dos casos de uso, a especificação de casos de uso deve ter sido revista pelo cliente, para verificar a sua consistência e completitude e , ainda, se a mesma atende às necessidades do cliente.

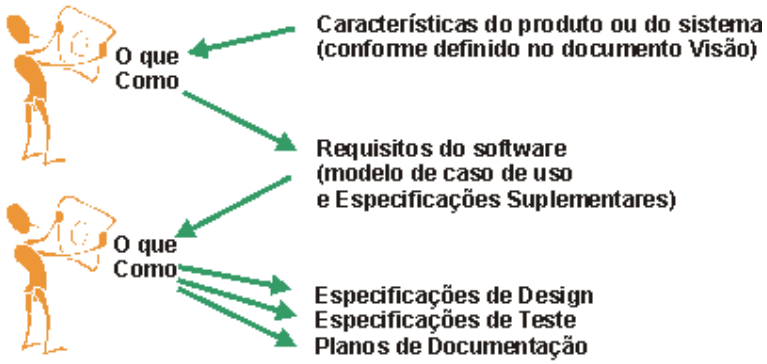


Figura 2.11: Fronteiras de um caso de uso (RUP, 2001)

Em um ambiente de desenvolvimento interativo, deve-se selecionar um subconjunto de casos de uso para ser detalhado em cada iteração (RUP, 2001). Esse detalhamento deverá evidenciar como o sistema interage com os atores, e como se comporta em cada passo do fluxo de eventos.

Há três motivos relevantes para se elaborar uma especificação de casos de uso:

- facilitar seu entendimento;
- particionar o comportamento comum descrito em muitos casos de uso; e
- propiciar sua manutenção.

A especificação de casos de uso poderá ser elaborada, quando se souber um pouco mais sobre o comportamento dos casos de uso e se ter descrito seu fluxo de eventos. Com isto, pode-se verificar se as decisões tomadas basearam-se em um entendimento bem preciso de seu comportamento (RUP, 2001).

Um modelo de *Especificação de Casos de Uso* é proposto no Apêndice A, a partir dos conceitos abordados neste capítulo, juntamente com um guia de utilização dessa especificação (Apêndice B).

Para exemplificar a utilização da Especificação de Casos de Uso proposta, são apresentados três casos de uso: “Manter Atendente” (Apêndice C), “Manter Usuário” (Apêndice D) e “Efetuar *Login*” (Apêndice E), de um sistema de *Call Center*.

2.11 Conclusão

Este capítulo apresentou os principais conceitos sobre casos de uso e propõe um modelo de especificação de casos de uso, apresentando orientações de como preenchê-lo. Isto poderá servir de base para utilização da técnica de estimativa de tamanho de software – a TUCP – proposta neste trabalho. Referida técnica depende fortemente que casos de uso sejam bem escritos, bem estruturados e tenham um nível de detalhamento apropriado.

O capítulo seguinte aborda o CMMI-SW (SEI, 2002), especificamente nas áreas de processo de planejamento e acompanhamento de projetos de software, por tratarem de questões pertinentes a estimativas de software, foco deste trabalho.

Capítulo **3**

O CMMI

Este capítulo apresenta o CMMI (Capability Maturity Model Integration) e as áreas de processos referentes a planejamento e acompanhamento de projetos de software que tratam mais diretamente de estimativas.

Ao longo dos anos 80 a indústria de software percebeu que apenas a utilização de ferramentas não significava, necessariamente, uma garantia de sucesso para um projeto de software. Os insucessos nos projetos, normalmente, não eram decorrentes da implantação de novas tecnologias e do uso de ferramentas, mas sim, da ausência de um processo de software disciplinado. Experiências demonstraram que o uso de novas tecnologias e ferramentas em um ambiente de desenvolvimento imaturo ou não disciplinado tem levado ao aumento dos problemas já existentes (ZAHRAN, 1998). Como resultado, a indústria de software tem enfrentado problemas de subestimativas de orçamento, atrasos no cronograma, dificuldade na medição do progresso do projeto, baixa qualidade do produto, falta de credibilidade no desempenho de TI e insatisfação do usuário (ABEL-HAMID, 1993), citado por (JURISON, 1999).

Diante desse contexto, o movimento de maturidade de processos surgiu como um segundo estágio na indústria de software, em resposta ao aumento dos insucessos em projetos. O objetivo de referido movimento não seria somente a definição de processos de software pelas organizações, mas, também, melhorar a gestão de projetos e providenciar ações gerenciais que pudessem contribuir para uma gestão mais efetiva e, conseqüentemente, melhorar seus índices (LAI, 1993; ZAHRAN, 1998).

Nos últimos anos, diversas normas, padrões e modelos têm sido desenvolvidos com o objetivo de definir, avaliar e melhorar a qualidade dos processos de software. Esses modelos e padrões fornecem às organizações de software um guia de como obter controle em seus processos, para desenvolver e manter software e como evoluir em direção a uma cultura de engenharia de software e excelência de gestão.

Neste contexto, o CMMI foi projetado para guiar as organizações de software no processo de seleção das estratégias de melhoria, determinando a maturidade atual do processo

e identificando as questões mais críticas para a qualidade e melhoria do processo de software (SEI, 2002). O CMMI foi proposto com base nos modelos CMM (PAULK *et al*., 1997) e na norma ISO/IEC 15504 (ISO/IEC, 2003).

3.1 CMM

O CMM é um *framework* que se caracteriza por uma melhoria de processo, voltado para organizações mais maduras. Uma organização pode usar o CMM para determinar sua classificação de maturidade do processo de software e então estabelecer prioridades visando melhorias (PAULK, 1993a). Segundo Pádua (2003) "maturidade significa um programa de melhoria contínua de processos, que não pode ser projetado de forma intuitiva; ele deve refletir o acervo de experiências dos profissionais e de cada organização".

Desde o início desta década, o CMM vem sendo implantado em muitas organizações *urbe et orbi*, objetivando a padronização e a melhoria de processos de desenvolvimento de software. Esse objetivo segue a tendência da globalização da economia mundial, onde a padronização da produção de software surge como um elemento estratégico, para a obtenção de novos mercados e seu posterior controle. O princípio fundamental desse modelo baseia-se na qualidade do produto de software alcançada através da melhoria da qualidade de seus processos (SANDERS *et al*., 1994).

O CMM é estruturado em cinco níveis em ordem crescente de maturidade, e quando a organização se encontra num certo nível, deve seguir atividades determinadas pelo modelo, para atingir o nível seguinte (HUMPHREY, 1991). São eles: nível 1 ou inicial, nível 2 ou repetível, nível 3 ou definido, nível 4 ou gerenciado, e nível 5 ou otimização.

A partir do nível 2 do CMM, são incluídos grupos de atividades chaves, KPAs (*key process areas*), que têm por objetivo segmentar e facilitar o trabalho de melhoria dos processos de software, em cada nível considerado (PAULK *et al*., 1997).

O CMM é conhecido pelo público como SW-CMM (ou CMM para software). Isto porque, na esteira de seu sucesso, diversos outros "CMMs" foram criados, procurando cobrir outras áreas de interesse. Assim surgiram os seguintes modelos (PAULK *et al*., 1997):

- *Software Acquisition CMM* (SA-CMM): avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de software desenvolvidos por terceiros.
- *Systems Engineering CMM* (SE-CMM): avalia a maturidade da organização em seus processos de engenharia de sistemas, concebidos como algo maior que o

software. Um "sistema" inclui o hardware, o software e quaisquer outros elementos que participam do produto completo.

- *Integrated Product Development CMM* (IPD-CMM): ainda mais abrangente que o SE-CMM, inclui também outros processos necessários à produção, e o suporte ao produto, tais como suporte ao usuário, processos de fabricação, etc.
- *People CMM* (P-CMM): avalia a maturidade da organização em seus processos de administração de recursos humanos, no que se refere a software, recrutamento e seleção de desenvolvedores, treinamento e desenvolvimento, remuneração, etc.

Usuários do CMM têm relatado que melhorias no produto e processo são alcançáveis pela perseverante padronização de processos com o uso das práticas base.

Em ambientes complexos de desenvolvimento de software, onde várias dessas práticas são empregadas, o uso coletivo de modelos individuais tem resultado em redundâncias, complexidade adicional, aumento de custos e, algumas vezes, discrepâncias. Para melhorar a eficiência do uso do CMM e aumentar o retorno do investimento foi desenvolvido o CMMI (Cap*ability Maturity Model Integration*), a fim de prover um único conjunto integrado de modelos (SEI, 2002).

Assim como o CMM, a norma ISO/IEC 15504 propõe-se ser um padrão para a avaliação e a melhoria de processos e determinação da capacitação de uma organização, com base nas melhores características de modelos de avaliação.

3.2 ISO/IEC 15504

A ISO realizou o estudo "Necessidades e Exigências para uma Norma de Avaliação de Processos de Software" em 1992. Referido estudo concluiu que era pertinente a elaboração de um padrão aplicável à melhoria de processos e à determinação da capacidade. Este padrão deveria considerar os métodos e normas já existentes (como, por exemplo, o CMM e a ISO 9001), abranger todos os processos de software e ser construído pelos especialistas, que já desenvolviam e trabalhavam com métodos e normas existentes na época.

Como resultado deste primeiro trabalho, a ISO iniciou em janeiro de 1993 o projeto SPICE (*Software Process Improvement and Capability Determination*) com o objetivo de produzir inicialmente um relatório técnico que fosse, ao mesmo tempo, mais geral e abrangente que os modelos existentes, e mais específico que a norma ISO 9001 (Salviano et al, 2001). Uma versão do SPICE foi aprovada em 1998 como Relatório Técnico e, em 2003, publicada a norma ISO/IEC 15504 (ISO/IEC, 2003).

O objetivo principal da norma SPICE é avaliar a capacitação da organização em cada processo e permitir a sua melhoria com o uso dos seguintes seis níveis de capacitação, que classificam os processos (ISO/IEC, 2003): nível 0 ou incompleto, nível 1 ou realizado, nível 2 ou gerenciado, nível 3 ou estabelecido, nível 4 ou previsível, e nível 5 ou otimizado.

A seguir, será mostrado um quadro comparativo dos modelos de capacidade e maturidade de software, em que se destacam os níveis de maturidade e objetivos de cada modelo, conforme a Tabela 3.1.

Tabela 3.1 – Comparativo de modelos de processos de software

CMM	SPICE (15504)	CMMI
Níveis de maturidade: 1. Inicial 2. Repetível 3. Definido 4. Gerenciado 5. Em Otimização	**Níveis de maturidade**: 0. Incompleto 1. Realizado 2. Gerenciado 3. Estabelecido 4. Previsível 5. Otimizado	**Níveis de maturidade**: 1. Inicial 2. Gerenciado 3. Definido 4. Gerenciado Quantitativamente 5. Otimizando
Objetivo: Auxiliar as organizações a aumentarem a maturidade de seu processo de produção de software através de um caminho evolutivo.	**Objetivo**: Normalização de orientações para a avaliação do processo de software, visando à melhoria contínua do processo e a determinação da capacitação.	**Objetivo**: Estabelecer um guia para melhorar o processo da organização e sua capacidade para gerenciar o desenvolvimento, a aquisição, e a manutenção de produtos ou serviços.

A ISO/IEC 15504, assim como o CMM, é uma das referências utilizadas pelo modelo CMMI, levando em conta as áreas de processo e níveis de maturidade estabelecidos (SEI, 2002). A seguir será apresentado o modelo CMMI.

3.3 CMMI

O CMMI foi projetado para guiar as organizações de software no processo de seleção das estratégias de melhoria, determinando a maturidade atual do processo e identificando as questões mais críticas para a qualidade e a melhoria do processo de software (CHRISSIS *et al*., 2003).

É um modelo para avaliação e melhoria da maturidade dos processos de uma organização, criado pelo *Software Engineering Institute*[1] (SEI) como integração e evolução dos seguintes modelos: SW-CMM - *Capability Maturity Model for Software*; SE-CMM - *System Engineering Capability Model*, e IPD-CMM - *Integrated Product Development* CMM. Embora estes modelos tenham tido sua utilidade, o uso de múltiplos modelos se mostrou

problemático. Em primeiro lugar, nem todos usavam a mesma terminologia, de modo que um mesmo conceito podia receber nomes diferentes em cada modelo, ou o mesmo termo significar coisas diferentes nos vários modelos. Além disso, a estrutura carecia de um formato padrão. Os modelos tinham diferentes números de níveis ou formas diferentes de avaliar o progresso. Um terceiro problema eram os altos custos de treinamento, avaliação e harmonização para as organizações que tentassem usar mais de um modelo.

Por todas estas razões, o SEI iniciou um projeto chamado CMMI (*Capability Maturity Model Integration*) baseado na experiência do uso do SW-CMM durante uma década, que serviu para identificar pontos em que os modelos poderiam ser melhorados. Concretamente, a primeira idéia, como o nome sugere, é integrar os diversos CMMs numa estrutura única, todos com a mesma terminologia, processos de avaliação e estrutura. Além disso, o projeto também se preocupou em tornar o CMM compatível com a norma ISO/IEC 15504, de modo que avaliações em um modelo fossem reconhecidas como equivalentes aos do outro e, naturalmente, incorporassem ao CMM as sugestões de melhoria surgidas ao longo dos anos de sua utilização.

O propósito do CMMI é estabelecer um guia para melhorar o processo da organização e sua capacidade para gerenciar o desenvolvimento, a aquisição e a manutenção de produtos ou serviços. Seu foco principal é construir ferramentas que possibilitem a melhoria dos processos utilizados no desenvolvimento e na manutenção de produtos e sistemas.

A principal mudança do SW-CMM em relação ao CMMI é que, neste último, existe a possibilidade de utilização de duas diferentes abordagens para a melhoria de processos. As duas abordagens são conhecidas como o "modelo contínuo" e o "modelo em estágios" (SEI, 2002). A seguir serão mostradas as duas representações do modelo CMMI.

3.3.1 Representação do CMMI

A representação por estágios é a mesma representação utilizada no SW-CMM. Refere-se a um conjunto de áreas de processo para definir um caminho de melhoria para a organização, descrito em termos de níveis de maturidade. Já a representação contínua é o enfoque utilizado no SE-CMM, no IPD-CMM e também na ISO/IEC 15504. Este enfoque permite que uma organização selecione uma área de processo específica e melhore com relação àquela área. A representação contínua usa níveis de capacidade para caracterizar melhoria relacionada a uma área de processo. Essas representações estão na Figura 3.1.

[1] O SEI é uma entidade mantida pelo governo norte-americano e patrocinada pelo Departamento de Defesa (US DoD), com o propósito de "auxiliar organizações a realizar melhorias mensuráveis em suas capacidades relativas à engenharia de software" (CHRISSIS *et al*., 2003).

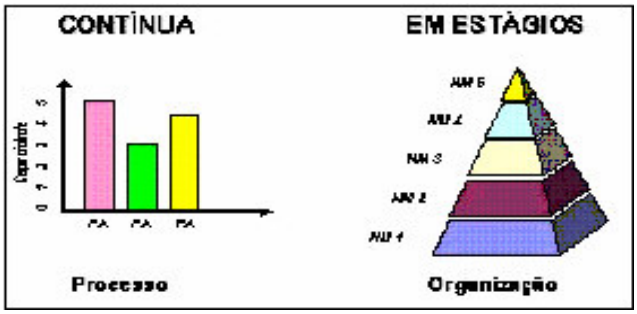


Figura 3.1 - Representação do CMMI

A melhor maneira de uma organização escolher o tipo a seguir, dependerá de como ela atua. Se uma organização já está familiarizada com o SW-CMM, a representação por estágios será a mais adequada para migrar para o CMMI. Essa representação também é a mais adequada se a organização necessita demonstrar externamente o seu nível de maturidade. Entretanto, não há obrigatoriedade na escolha de uma representação em detrimento da outra.

Mais de 80% do conteúdo das duas representações são comuns e elas oferecem resultados equivalentes. Raramente as organizações implementam uma representação exatamente como ela é prescrita. Por exemplo, uma organização pode escolher a representação em estágios para implementar o nível 2, mas incluir uma ou duas áreas de processo de nível 3 em seu plano de melhoria.

Uma outra possibilidade é uma organização escolher a representação contínua para guiar internamente o seu processo de melhoria e, no momento de realizar a avaliação, escolher a representação em estágios (CHRISSIS *et al*., 2003). Caso uma organização não esteja familiarizada com nenhuma destas duas representações, poderá implementar qualquer uma das duas, mas será necessário verificar qual o objetivo que esta organização quer atingir ao escolher uma das representações (CHRISSIS *et al*., 2003).

Cada representação tem suas vantagens em relação à outra. Algumas organizações utilizam ambas as representações para guiar necessidades particulares em vários momentos durante programas de melhoria. A seguir, serão apresentadas as vantagens e as desvantagens de cada representação.

Representação Contínua

Segundo o SEI (2002), a representação contínua oferece uma proposta mais flexível para o processo de melhoria. Através dessa representação, uma organização pode escolher melhorar o seu desempenho em um único ponto do processo relacionado a um determinado problema ou pode trabalhar em diversas áreas que estão próximas aos objetivos de negócio da organização.

A representação contínua permite também que uma organização melhore seus processos em categorias diferentes. No entanto, há algumas limitações com relação à escolha por causa das dependências entre algumas áreas de processo.

Os níveis de capacidade usados para medir a melhoria contínua para cada área de processo vão desde a inexistência de um processo até um processo otimizado. Por exemplo, uma organização pode desejar alcançar o nível 2 de uma determinada área de processo e um nível 4 de uma outra área. Quanto uma organização atinge um nível da capacidade, ela poderá decidir se amplia seu escopo e cria o mesmo nível de maturidade para outras áreas de processo.

Representação por Estágio

A representação por estágio oferece uma maneira sistemática e estruturada de propor uma melhoria do processo seguindo uma etapa de cada vez. Atingir cada estágio assegura que uma melhoria adequada esteja colocada como uma base para o estágio seguinte. As áreas de processo são organizadas por níveis de maturidade onde, a partir de cada nível, fornecem uma gama de fundamentos para a melhoria contínua do processo. A representação por estágio prescreve um método para implementar cada área de processo de acordo com o nível de maturidade. Cada nível de maturidade contempla diferentes componentes no processo de software, resultando em um crescimento na capacidade de processo da organização (SEI, 2002).

Se uma organização tem o foco no desenvolvimento de produtos e quer melhorar os processos para toda a organização, deverá optar por aplicar a representação por estágio. A representação por estágio ajudará a organização a selecionar os processos críticos para focar sobre a melhoria. Se a mesma organização escolher melhorar os processos para o desenvolvimento do produto, então para este caso a representação contínua deve ser aplicada. Ambas as abordagens são válidas. A escolha da representação do CMMI para uma organização deve levar em consideração os objetivos do negócio e se estes objetivos estão alinhados com alguma das duas representações, conforme Tabela 3.2.

Tabela 3.2 - Comparação entre as duas representações do CMMI

Contínua	Estágio
▪ Melhorar desempenho em um processo único	▪ Enfoque de melhoria do processo de forma sistêmica e estruturada
▪ Melhorar desempenho em várias áreas alinhadas aos objetivos de negócio da organização	▪ Atingir cada um dos estágios garante a base fundamentada necessária para o próximo estágio
▪ Níveis de capacidade utilizados para medir as melhorias	▪ PA´s organizadas em níveis de maturidade
▪ Melhorar diferentes processos com diferentes classificações	▪ Permite a organização ter um caminho evolutivo pré-definido para melhoria
▪ Necessário conhecimento das dependências e interações entre áreas de processos (PA)	▪ Provê a migração mais fácil do SW-CMM para o CMMI
▪ Apropriado para quem sabe qual processo deve ser melhorado	▪ Apropriado para quem não sabe como iniciar um processo de melhoria ou qual processo deve ser prioridade
▪ Alinhado com a ISO/IEC 15.504 devido a organização idêntica das PA´s.	▪ Possui grande número de casos de estudo e dados históricos de práticas bem sucedidas
	▪ Recomendado para que está atuando com SW-CMM

A seguir serão apresentados os componentes do modelo CMMI que são comuns para ambas as abordagens apresentadas acima.

3.3.2 Componentes do Modelo

Ambas as representações (por estágio e contínua) do CMMI apresentam os seguintes componentes: *áreas de processo* (PA), *objetivos específicos* (SG), *práticas especificas* (SP), *objetivos genéricos* (GG), *práticas genéricas* (GP), *produtos típicos de trabalho*, *subpráticas*, *notas*, *amplificações das disciplinas*, *elaborações das práticas genéricas*, e *referências*, conforme Figura 3.2 e Figura 3.3 (SEI, 2002).

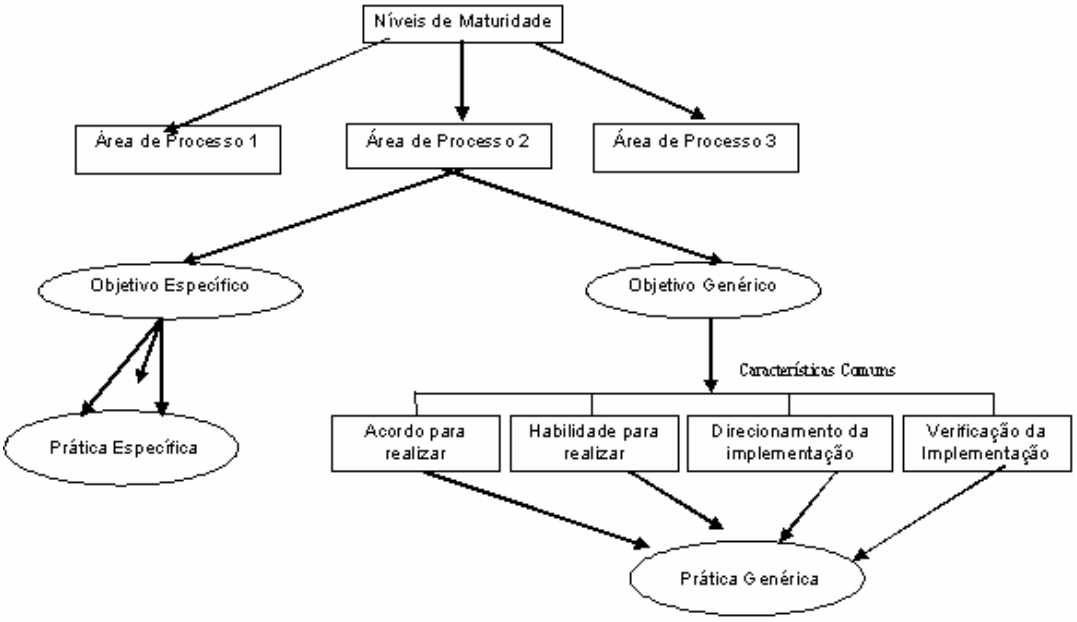


Figura 3.2 – Componentes da representação por estágio

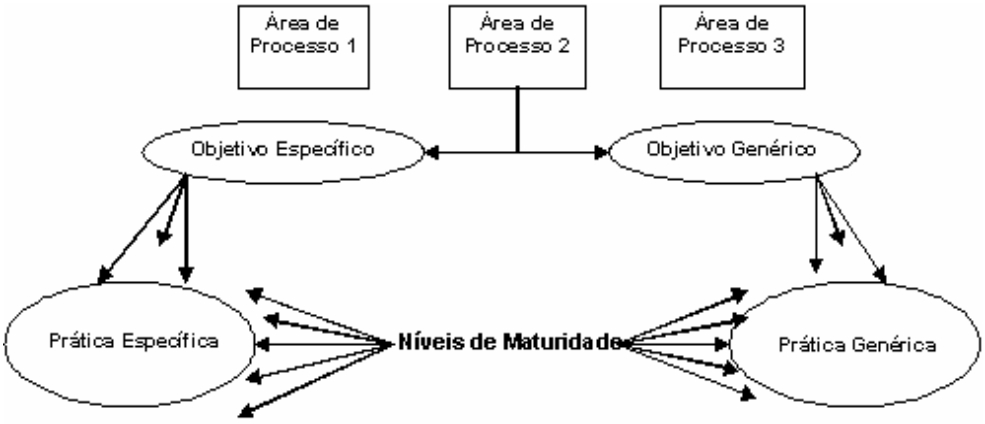


Figura 3.3 – Componentes da representação contínua

Área de Processo (PA)

Uma área de processo é um *cluster* de práticas associadas a uma determinada área que, quando executada coletivamente, satisfaz a um conjunto de objetivos considerados importantes para uma melhoria significativa nessa área (SEI, 2002).

As áreas de processos podem ser agrupadas em quatro categorias: (*i*) *Gerenciamento de Processo*; (*ii*) *Gerenciamento de Projeto*; (*iii*) *Engenharia*; e (*iv*) *Suporte*. Neste trabalho, a categoria de *gerenciamento de projetos* terá uma maior importância, já que tratam do *planejamento de projeto*, e do *monitoramento e controle de projeto*, que se relacionam diretamente com estimativas de software.

Objetivos Específicos (SG)

Os *objetivos específicos* tratam de características únicas que descrevem o que deve ser implementado, para satisfazer uma determinada área de processo. São componentes requeridos do modelo e usados em avaliações para verificar se uma determinada área de processo foi satisfeita.

Práticas Específicas (SP)

As *práticas específicas* são atividades consideradas importantes para atingir um dado objetivo específico. Descrevem as atividades esperadas para atingir o resultado de um objetivo específico.

Objetivos Genéricos (GG)

Os *objetivos genéricos* são chamados assim, porque a mesma indicação do objetivo aparece em diversas áreas de processo. A realização de um objetivo genérico de uma área de processo significa melhorar o controle no planejamento e na execução dos processos associados àquela área de processo. Assim, indica se este processo é eficaz, repetível e permanente. Os objetivos genéricos são componentes requeridos pelo modelo e são usados nas avaliações, para determinar se uma área de processo está satisfeita.

Práticas Genéricas (GP)

As *práticas genéricas* promovem a institucionalização, assegurando que os processos associados à área de processo sejam eficazes, repetíveis e permanentes. As práticas genéricas são categorizadas por: *objetivos genéricos*, e *características comuns*, sendo componentes esperados do CMMI.

Quatro características comuns organizam as práticas genéricas de cada área de processo:

- *Acordo para Realizar* (CO): grupos de práticas genéricas relacionadas à criação das políticas e da garantia do patrocínio.
- *Habilidade para Realizar* (AB): grupos de práticas genéricas relacionadas à garantia de o projeto ou à organização dispor de recursos necessários.

- *Direcionamento para Implementação* (DI): grupos de práticas genéricas relacionadas ao gerenciamento da execução dos processos, à integridade dos produtos de trabalho, e ao envolvimento dos *stakeholders*[2] relevantes.
- *Verificação da Implementação* (VE): grupos de práticas genéricas relacionadas à revisão pela alta gerência e avaliação da conformidade com relação ao processo descrito, aos procedimentos e padrões.

Os componentes do CMMI podem se classificados como "requerido", "esperado" e "informativo". Os itens "requeridos" são os mais importantes e essenciais para o modelo e para o entendimento do que é necessário na melhoria do processo, e na demonstração de conformidade com o modelo. Os itens "esperados", que embora possam não estar presentes na organização, por não serem essenciais, são fortes indicadores de que um item "requerido" foi alcançado. Por fim, tem-se o material "informativo" que constitui um guia para a implementação do modelo.

Os únicos componentes requeridos do CMMI são os objetivos. Assim, quando um objetivo é específico de uma determinada área de processo, é chamada de objetivo específico. Quando um objetivo pode ser aplicado em todas as áreas de processo, é chamado de objetivo genérico. As práticas são componentes esperados do CMMI. Cada prática está mapeada para apenas um objetivo. Quando uma prática é específica de uma área de processo, é chamada de prática específica. Por conseguinte, quando uma prática pode ser aplicada a todas as áreas de processo, é chamada de prática genérica (AHERN, 2001, CHRISSIS *et al.*, 2003).

A representação por estágio organiza suas áreas em cinco níveis para suportar e conduzir a melhoria do processo. Além disso, agrupa as áreas de processos por nível de maturidade, indicando que um determinado conjunto de áreas de processos deve ser implementado para que um nível de maturidade seja atingido, tal como no SW-CMM.

Cada área de processo é composta por objetivos genéricos e objetivos específicos. Apenas a representação por estágio utiliza as quatro características comuns para organizar as práticas genéricas (Figura 3.2).

Como ilustrado na Figura 3.3, na representação contínua, os objetivos específicos organizam as práticas específicas e os objetivos genéricos organizam as práticas genéricas. Os objetivos específicos e as práticas específicas aplicam-se às áreas de processo individuais.

[2] *Stakeholders* é um grupo ou um indivíduo afetado ou de alguma maneira responsável por um resultado de uma empresa ou tarefa. Um *stakeholder* relevante é usado para designar um *stakeholder* que é identificado pelo envolvimento em atividades específicas e é incluído em um plano apropriado. (CMMI, 2003).

Os objetivos genéricos e as práticas genéricas aplicam-se às diversas áreas de processo. Os objetivos genéricos e as práticas genéricas definem uma seqüência dos níveis da capacidade, que representam as melhorias com relação à execução e à eficácia dos processos que se escolheu melhorar.

3.3.3 Níveis de Maturidade do CMMI-SW

O nível de maturidade de uma organização pode predizer o desempenho futuro de uma organização dentro de uma disciplina ou de um conjunto de disciplinas. Experiências têm mostrado que as organizações fazem seu melhor, quando focalizam seus esforços na melhoria do processo sobre um número gerenciável de áreas de processo, na qual requerem um esforço cada vez mais apurado com a melhoria da organização.

Um nível de maturidade é um platô evolutivo definido no processo de melhoria. Cada nível de maturidade estabiliza uma parte importante dos processos organizacionais.

A representação por estágios do CMMI é organizada em cinco níveis de maturidade, que estabelecem fundamentos sucessivos para a continua melhoria do processo: nível 1 – Inicial, nível 2 – Gerenciado, nível 3 – Definido, nível 4 – Gerenciado Quantitativamente, e nível 5 – Em otimização.

Já a representação contínua do CMMI é organizada em seis níveis de capacidade: nível 0 – Incompleto, nível 1 – Realizado, nível 2 – Gerenciado, nível 3 – Definido, nível 4 – Gerenciado Quantitativamente, e nível 5 – Otimizando.

Os níveis de maturidade consistem em um conjunto predefinido de áreas de processo. Esses níveis são medidos através da execução dos objetivos específicos e genéricos, que se aplicam a cada conjunto predefinido nas áreas de processo. A seguir são descritas as características de cada nível de maturidade da representação por estágio.

Nível de Maturidade 1: Inicial

O nível de maturidade 1 é caracterizado como *ad hoc* e caótico. A organização normalmente não fornece um ambiente estável. O sucesso desta organização depende da competência e heroísmo das pessoas que a compõem. Raros são os processos estáveis em evidência e o desempenho só pode ser previsto através de habilidades individuais, e não por meio da capacidade da organização. É caracterizado por ter seus compromissos não cumpridos, processos abandonados em períodos de crises, não sendo possível repetir os sucessos do passado.

Nível de Maturidade 2: Gerenciado

Uma organização consegue atingir o nível de maturidade gerenciado, quando todos os objetivos específicos e genéricos das áreas de processo forem atingidos. Em outras palavras, os projetos da organização asseguram que os requisitos são gerenciados e que os processos são planejados, executados, medidos e controlados.

Os processos referentes ao nível 2 ajudam a garantir que práticas existentes são preservadas nos projetos em momentos de estresse. Quando essas práticas estão sendo aplicadas adequadamente, projetos são executados e realizados de acordo com planos documentados. Os processos, produtos de trabalho e serviços são gerenciados e controlados. O *status* dos produtos de trabalho e dos serviços é visível para o gerenciamento nos pontos de controle dos projetos (marcos e finalização de tarefas).

Os compromissos estabelecidos entre os *stakeholders* relevantes são revisados, quando necessário. Os produtos de trabalhos, serviços e processos são revisados com os *stakeholders* para verificar se requisitos, padrões e objetivos estão sendo cumpridos.

Nesse nível pode haver diferenças na maneira de como os projetos aplicam os processos. Suas as áreas de processos são as seguintes: (*i*) *Gerenciamento de Requisitos*, (*ii*) *Planejamento do Projeto*, (*iii*) *Monitoração e Controle do Projeto*, (*iv*) *Gerenciamento de Configuração*, (*v*) *Garantia da Qualidade do Processo e do Produto*, (*vi*) *Gerenciamento de Acordos com Fornecedores*, e (*vii*) *Medição e Análise*.

Nível de Maturidade 3: Definido

Uma organização consegue atingir o nível de maturidade definido quando todos os objetivos específicos e genéricos das áreas de processo dos níveis 2 e 3 foram atingidos. Nesse nível, os processos estão bem caracterizados, compreendidos e descritos como padrões, procedimentos, ferramentas e métodos.

O conjunto de processos padrão da organização, que é base para o nível 3, é estabelecido e melhorado ao longo do tempo. Os processos são comuns a toda a organização para estabelecer uma consistência e podem ser adequados às necessidades de cada projeto seguindo guias de adaptação. Os projetos estabelecem seus processos definidos por meio da adaptação de um conjunto de processos padrão da organização através de um guia de adaptação. A gerência estabelece objetivos de processo com base nesse conjunto de processos padrão e garante que estes objetivos são apropriadamente tratados.

No nível de maturidade 2, os padrões, as descrições dos processos e procedimentos podem ser completamente diferentes em cada instância específica do processo (por exemplo,

em um projeto em particular). No nível de maturidade 3, os padrões, as descrições de processo e os procedimentos para um projeto são adaptados de um conjunto padrão de processos da organização para servir um projeto em particular ou uma unidade organizacional. O conjunto padrão de processos organizacionais inclui os processos destinados para os níveis 2 e 3. Os processos devem ser seguidos por toda a organização; a única exceção são os projetos onde seus processos são adaptados segundo um guia de adaptação.

Outro ponto com relação ao nível 3 é que os processos são descritos com mais detalhes e são mais rigorosos do que o nível 2. No nível 3, os processos são gerenciados de maneira mais proativa utilizando o conhecimento dos inter-relacionamentos das atividades dos processos e do detalhamento das medições de cada um deles, dos produtos de trabalho e dos serviços.

No nível de maturidade 3 as áreas de processos são as seguintes: (*i*) *Desenvolvimento de Requisitos*, (*ii*) *Solução Técnica*, (*iii*) *Integração do Produto*, (*iv*) *Verificação*, (*v*) *Validação*, (*vi*) *Treinamento Organizacional*, (*vii*) *Foco no Processo Organizacional*, (*viii*) *Definição do Processo Organizacional*, (*ix*) *Gerência Integrada do Projeto*, (*x*) *Gerência de Riscos*, e (*xi*) *Análise e Resolução da Decisão.*

Nível de Maturidade 4: Gerenciado Quantitativamente

Uma organização consegue atingir o nível de maturidade gerenciado quantitativamente, quando todos os objetivos específicos das áreas de processo dos níveis 2, 3 e 4 e todos os objetivos genéricos dos níveis 2 e 3 forem atingidos. No nível 4, os subprocessos[3] são selecionados de modo que contribuam significativamente para o desempenho de todo o processo. Esses subprocessos selecionados são controlados usando técnicas estatísticas e outras quantitativas.

Os objetivos quantitativos para a qualidade e a execução dos processos são estabelecidos e utilizados como critérios no gerenciamento dos processos. São baseados nas necessidades de clientes, de usuários, da organização e dos implementadores e executores dos processos. A qualidade e a execução dos processos são compreendidas em termos estatísticos e gerenciadas através do ciclo de vida do processo.

Para estes processos, medições detalhadas das execuções dos processos são coletadas e analisadas estatisticamente. *Causas especiais de variação do processo*[4] são identificadas e,

[3] Um processo, que é parte de um processo maior (SEI, 2002).
[4] Uma causa de um defeito que é específico de uma circunstância passageira e que não é parte inerente de um processo (SEI, 2002).

quando apropriadas, as fontes dessas causas especiais são corrigidas para prevenir ocorrências futuras.

As medições da qualidade e da execução dos processos são incorporadas em um repositório organizacional de medições, para servir de suporte para decisões a serem tomadas no futuro, as quais se devem basear em fatos.

Uma distinção entre os níveis 3 e 4 é a previsibilidade da execução dos processos. No nível 4, a execução dos processos é controlada utilizando técnicas quantitativas e estatísticas e é quantitativamente previsível, enquanto que, no nível 3, os processos são somente quantitativamente previsíveis.

No nível de maturidade 4, as áreas de processos são as seguintes: (*i*) *Desempenho do Processo Organizacional*, e (*ii*) *Gerência Quantitativa do Projeto*.

Nível de Maturidade 5: Em otimização

Uma organização consegue atingir o nível de maturidade otimizando, quando todos os objetivos específicos das áreas de projeto dos níveis 2, 3, 4 e 5, e os todos os objetivos genéricos dos níveis 2 e 3 são atingidos. Os processos são continuamente melhorados, são baseados no entendimento quantitativo das *causas comuns de variação do processo*[5].

O nível de maturidade 5 foca a melhoria contínua do processo através de melhorias incrementais nos processos e em inovações tecnológicas. Os efeitos da implantação dos processos de melhorias são medidos e avaliados diante dos objetivos quantitativos do processo de melhoria. Tanto os processos definidos, quanto o conjunto de processos padrão da organização são alvo das medições das atividades de melhoria.

Os processos de melhorias para tratar as causas comuns de variação do processo e as melhorias mensuráveis dos processos organizacionais são identificados, avaliados e implantados. As melhorias são selecionadas baseadas nos entendimentos quantitativos de suas esperadas contribuições, para atingir os objetivos das melhorias dos processos organizacionais versus o custo e o impacto para a organização. A execução dos processos organizacionais é continuamente aperfeiçoada.

A otimização dos ativos organizacionais e dos novos processos dependem da participação de todos os envolvidos e sua capacitação está associada aos valores de negócio e objetivos da organização. A melhoria dos processos é parte inerente de todos os envolvidos, tendo como resultado um ciclo de melhoria contínua.

[5] Uma variação do processo existente decorrente de uma interação normal e esperada entre componentes de um processo (SEI, 2002).

Uma distinção entre o nível 4 e o nível 5 está no tipo do tratamento da variação do processo. No nível 4, os processos dizem respeito ao tratamento das causas especiais de variação do processo, e fornecem resultados estatísticos previsíveis. Embora os processos possam produzir resultados previsíveis, estes resultados podem ser insuficientes para atingir os objetivos estabelecidos.

No nível 5, os processos dizem respeito ao tratamento das causas comuns de variação do processo e da mudança do processo (isto é, afastando-se da execução do processo pretendido), para melhorar o desempenho do processo (manter a previsibilidade estatística), atingindo os objetivos quantitativos estabelecidos do processo de melhoria.

No nível de maturidade 5, as áreas de processos são as seguintes: (*i*) *Implantação e Inovação Organizacional*, e (*ii*) *Análise e Resolução de Causas*.

A seguir, serão detalhadas as áreas de processo "*Planejamento de Projeto*" e "*Monitoração e Controle do Projeto*" referentes ao nível de maturidade 2 do modelo CMMI-SW com a representação por estágio. Estas áreas de processo são focos principais deste trabalho.

3.4 Planejamento de Projeto

O propósito da área de processo *Planejamento de Projeto* do nível 2 é estabelecer e manter planos que definam as atividades do projeto, isto é, estabelecer planos razoáveis para a execução das atividades de engenharia de software e para a gestão do projeto de software. Essa área compreende:

- Elaborar um plano de projeto.
- Possibilitar a interação entre os *stakeholders* apropriadamente.
- Garantir os compromissos assumidos no plano do projeto.
- Manter o plano de projeto atualizado.

O planejamento se inicia com os requisitos que são definidos para o produto e projeto. Com isto devem-se estabelecer o trabalho a ser realizado e as restrições e metas que definem e limitam o escopo do projeto.

O planejamento inclui passos para estimar os atributos dos produtos de trabalho e de tarefas, determinar recursos necessários, elaborar um cronograma, identificar e avaliar riscos e negociar compromissos. Devido a iterações desses passos, pode ser necessário estabelecer um plano para o projeto. Esse plano fornece a base para a execução e o controle das atividades do projeto e explicita os compromissos com o cliente do projeto.

O plano de projeto é comumente revisado e atualizado com o progresso do projeto para tratar as mudanças de requisitos e compromissos, estimativas imprecisas, ações corretivas e alterações no processo. Nesta área de processo, as práticas específicas descrevem tanto o planejamento, quanto o replanejamento de projeto.

Os *objetivos específicos* e as práticas relacionadas a cada objetivo específico são as seguintes:

- Estabelecer Estimativas:
  - Estimar o escopo do projeto;
  - Estabelecer as estimativas dos *produtos de trabalho e tarefas*[6];
  - Definir ciclo de vida do projeto;
  - Determinar estimativas de esforço e custo.
- Desenvolver um plano de projeto:
  - Estabelecer um orçamento e cronograma;
  - Identificar os riscos do projeto;
  - Planejar o gerenciamento dos dados;
  - Planejar os recursos do projeto;
  - Planejar conhecimento e habilidades necessárias;
  - Planejar o envolvimento dos *stakeholders*;
  - Estabelecer o plano do projeto.
- Obter comprometimento com o plano:
  - Revisar planos que afetam o projeto;
  - Associar recursos e tarefas;
  - Obter o comprometimento com o plano.

A seguir, cada objetivo específico apresentado acima será detalhado.

Estabelecer Estimativas

O objetivo específico "*Estabelecer Estimativas*" descreve o estabelecimento e manutenção das estimativas dos parâmetros do planejamento do projeto. Os parâmetros do planejamento do projeto incluem todas as informações indispensáveis para o projeto executar

[6] Produto de trabalho e tarefas são características dos produtos, serviços e tarefas do projeto utilizado para ajudar na estimativa do produto de trabalho. Estas características incluem os seguintes itens: tamanho, complexidade, peso, forma, adaptação ou função, que são tipicamente utilizados como uma entrada para derivar estimativas de outros projetos e recursos (por exemplo: esforço, custo, cronograma). Produto de trabalho significa qualquer artefato produzido por um processo. Estes artefatos podem ser: arquivos, documentos, partes de um produto, serviços, processos, especificações ou uma nota fiscal (SEI, 2002).

o planejamento: alocar as atividades, os recursos (materiais e humanos), gerenciar as atividades, coordenar, reportar resultados e orçar custos.

As estimativas dos parâmetros do planejamento devem ter uma base segura, para assegurar que qualquer plano baseado nestas estimativas sejam capaz de garantir os objetivos do projeto. Em outras palavras, as estimativas do projeto, que são base para o seu planejamento, devem ser confiáveis, pois será a partir delas que todos os compromissos assumidos no plano poderão ser realizados. Os fatores que devem ser considerados para estes parâmetros são os seguintes:

- Requisitos do projeto, incluindo os requisitos do produto, os requisitos impostos pela organização, os requisitos exigidos pelo cliente e outros requisitos que impactem o projeto.
- Escopo do projeto.
- Tarefas e produtos de trabalho identificados.
- Abordagem técnica[7] .
- Modelo do ciclo de vida do projeto selecionado (por exemplo: cascata, incremental, espiral, etc.).
- Atributos dos produtos de trabalho e tarefas (por exemplo: tamanho e complexidade).
- Prazo.
- Modelos ou dados históricos para converter os atributos dos produtos de trabalho e tarefas em horas trabalhadas e em custos.
- Metodologia (modelos, dados, algoritmos) usada para determinar materiais necessários, habilidades, esforço e custo para o projeto.

As estimativas e os dados de suporte são necessários para análise dos *stakeholders,* para obter o comprometimento do plano e permitir que sejam realizadas as devidas manutenções com o progresso do projeto.

Referido objetivo específico é ponto motivador deste trabalho, pois o modelo pede que sejam estabelecidas estimativas como base formal para o planejamento, a partir de quatro práticas específicas: (*i*) estimar o escopo do projeto, (*ii*) estabelecer as estimativas dos

[7] A abordagem técnica define uma estratégia de alto nível para o desenvolvimento dos produtos. Isto inclui decisões sobre as características arquiteturais, como distribuído ou cliente-servidor; tecnologias novas ou já estabelecidas para serem aplicadas, como robótica e inteligência artificial; uma profundidade das funcionalidades esperadas para o produto final, como segurança e ergonomia (SEI, 2002).

produtos de trabalho e tarefas[8], (*iii*) definir ciclo de vida do projeto e (*iv*) determinar estimativas de esforço e custo.

Na prática específica "*Estimar o escopo do projeto*", é esperado que se estabeleça uma estrutura de divisão de trabalho (WBS) de alto nível, para estimar o escopo do projeto. Inicialmente, um alto nível da WBS pode servir como estrutura primária para a elaboração das estimativas. A elaboração da WBS divide todo o projeto em um conjunto de componentes gerenciáveis interconectados, o que possibilita uma estimativa mais precisa.

A WBS é tipicamente uma estrutura orientada para produto, que provê um esquema para identificar e organizar unidades de trabalho lógico para serem gerenciadas, que são chamadas de pacotes de trabalho (*work packages*). A WBS fornece uma referência e um mecanismo organizacional para determinar o esforço, o prazo e a responsabilidade. É utilizada como um *framework* base na elaboração do plano de projeto, organização das atividades e no controle do trabalho feito no projeto.

Os possíveis produtos típicos de trabalho esperados nesta prática são:

- Descrição das tarefas.
- Descrição dos pacotes de trabalho (*work packages*).
- WBS.

Na prática específica "*Estabelecer Estimativas dos Produtos de Trabalho e das Tarefas*", é esperado que sejam estabelecidas e mantidas as estimativas dos atributos dos produtos de trabalho e tarefas. Entende-se como atributos o tamanho e a complexidade de um produto de trabalho, por exemplo. Segundo CMMI (SEI, 2002), o tamanho é a principal entrada de muitos modelos utilizados para estimar esforço, custos e prazo. Os modelos para os cálculos das estimativas também podem ser baseados nas seguintes entradas: conectividade, complexidade e estrutura.

O modelo CMMI apenas sugere através de exemplos quais os tipos de produtos de trabalho a serem desenvolvidos com relação às estimativas de tamanho. Alguns exemplos deles são: (*i*) produtos de trabalho entregáveis e não entregáveis; (*ii*) documentação; e (*iii*) software operacional e de suporte. Assim, para que um projeto seja mais bem estimado, seus produtos de trabalho devem ser decompostos a um nível de granularidade necessária para se alcançar os objetivos da estimativa.

[8] Produto de trabalho e tarefas são características dos produtos, serviços e tarefas do projeto utilizado para ajudar na estimativa do produto de trabalho. Estas características incluem os seguintes itens: tamanho, complexidade, peso, forma, adaptação ou função. Eles são tipicamente utilizados como entrada para derivar estimativas de outros projetos e recursos (por exemplo: esforço, custo, cronograma). Produto de trabalho significa qualquer artefato produzido por um processo. Estes artefatos podem ser: arquivos, documentos, partes de um produto, serviços, processos, especificações ou uma nota fiscal (SEI, 2002).

Um ponto importante observado no modelo é que as estimativas devem ser consistentes com os requisitos do projeto para determinar esforço, custo e prazo do projeto. Um nível relativo de dificuldade ou complexidade deve ser associado a cada atributo do tamanho.

Métodos apropriados para determinar os atributos dos produtos de trabalho e tarefas serão utilizados para estimar os meios requeridos. Os métodos para determinar tamanho e complexidade devem ser baseados em modelos validados e dados históricos. Os métodos para determinar os atributos envolvem tanto o entendimento do relacionamento das características de produto, quanto o progresso dos atributos. Exemplos de métodos: número de portas lógicas para projeto integrado do circuito, linhas de código, pontos por função, ou pontos de caso de uso para o software, número ou complexidade dos requisitos para engenharia do sistema, etc.

O modelo também não especifica o tipo de medição que deve ser aplicada, nem a granularidade em que os produtos de trabalho devem ser decompostos. O CMMI (SEI, 2002) dá os seguintes exemplos de estimativas de tamanho a serem utilizadas: número de funções, pontos por função, linhas de código, número de classes e objetos, número de requisitos, números de interfaces, número de páginas, número de entradas e saídas, número de itens de riscos técnicos, volume de dados, etc.

O modelo sugere que as estimativas para os produtos de trabalho e tarefas sejam elaboradas, devendo também ser levadas em conta as seguintes estimativas: horas trabalhadas, maquinário a ser utilizado e métodos que serão requeridos pelo projeto.

Na prática específica "*Determinar Estimativas de Esforço e Custo*", é esperado que as estimativas de esforço e custo para os produtos de trabalho e tarefas sejam baseadas em uma estimativa confiável.

Estimativas de esforço e de custo são geralmente baseadas em resultados de análises, utilizando modelos ou dados históricos aplicados para tamanho, atividade e outro parâmetro de planejamento. Confiança nestas estimativas é baseada em uma análise criteriosa do modelo selecionado e a natureza dos dados a serem utilizados para efetuar todo o planejamento do projeto. As estimativas devem ser seguras o suficiente, para que os objetivos sejam atingidos dentro do esperado, baseado no planejamento.

Pode acontecer que os dados históricos não estejam disponíveis por ser um projeto com funcionalidades nunca implementadas pela organização ou os dados não estarem disponíveis para uma tarefa que não se adapte ao modelo. Esforços nunca implementados anteriormente são mais arriscados para o projeto, requerem mais pesquisas para desenvolver uma base aceitável de estimativa, além de um gerenciamento mais restrito. A singularidade de

um projeto deve ser documentada, quando utilizar os modelos para garantir um entendimento comum de quaisquer suposições realizadas em estágios iniciais de planejamento.

Nesta prática específica, deve ser realizada a coleta dos modelos ou dados históricos, que serão utilizados para transformar os atributos dos produtos de trabalho e tarefas em estimativas de esforço e custos. Além disso, incluir a infra-estrutura de suporte necessária, quando estimar esforço e custo, ou seja, levantar os recursos computacionais críticos, ambientes de teste necessários, ambientes de desenvolvimentos, os periféricos, dentre outros. Devem também ser levados em conta para efeitos de esforços e custos os seguintes pontos: riscos, capacitação dos recursos, viagens, horas extras, despesas indiretas.

Na prática específica "*Definir Ciclo de Vida do Projeto*", devem ser determinadas as etapas do ciclo de vida do projeto com base no esforço planejado. O ciclo de vida do projeto consiste de etapas, que necessitam ser definidas de acordo com o escopo dos requisitos, as estimativas para os recursos e o tipo de projeto.

O entendimento do ciclo de vida do projeto é crucial na determinação do escopo do planejamento e no sincronismo do planejamento inicial, bem como o sincronismo e critérios (marcos críticos) para o replanejamento.

Desenvolver um plano de projeto

O objetivo específico "*Desenvolver um plano de projeto*" exige que um plano de projeto seja estabelecido e mantido como base para gerenciar o projeto. Um plano de projeto é um documento formal usado para gerenciar e controlar sua execução, baseado nos requisitos e nas estimativas estabelecidas. O plano do projeto deve considerar todas as fases do ciclo de vida do projeto. O planejamento deve garantir que todos os planos, que afetam o projeto, sejam consistentes com o plano de projeto.

Na prática específica "*Estabelecer custo e prazo*", é esperado que o orçamento e o prazo sejam baseados nas estimativas elaboradas e garantam que a alocação de custo, a complexidade das tarefas e as dependências entre as atividades do projeto sejam tratadas apropriadamente.

Na prática específica "*Identificar Riscos do Projeto*", é esperado que os riscos identificados e analisados dêem apoio ao planejamento do projeto. Essa prática específica deve ser estendida a todos os planos que afetam o projeto, para assegurar uma comunicação apropriada entre todos os *stakeholders* relevantes sobre os riscos identificados.

Na prática específica "*Planejar a gerência de dados*", espera-se que seja realizado um planejamento sobre os dados do projeto. Os dados são as várias maneiras de documentação

requerida para suportar um programa de todas as áreas, como, por exemplo: administrativo, financeiro, engenharia, logística, configuração, qualidade, segurança, desenvolvimento, aquisição, etc. Os dados podem ser descritos e armazenados de diversas maneiras, podendo ser internos ao projeto ou não.

A razão para armazenar cada documento deve ser clara. Esta tarefa inclui a análise e a verificação dos produtos de trabalho entregáveis ou não, dados exigidos em contrato ou não e dados do fornecedor. Freqüentemente, dados são coletados sem um claro entendimento e como devem ser utilizados. Por serem caros, devem ser armazenados somente quando necessários.

Na prática específica "*Planejar recursos do projeto*", espera-se que sejam planejados os recursos necessários para executar o projeto. Os recursos do projeto (equipamento, material, mão-de-obra, componentes) necessários para executar as atividades devem ser definidos com base nas estimativas iniciais e em informações adicionais fornecidas, que podem ser obtidas através da expansão da WBS usada para gerenciar o projeto. Os recursos devem ser baseados no tamanho e no escopo definido para o projeto.

Na prática específica "*Planejar conhecimentos e habilidades necessárias*", espera-se que sejam planejados treinamentos para os envolvidos com o projeto, a fim de que sejam adquiridos conhecimentos ou habilidades para os papéis que lhes serão designados. Estes treinamentos poderão ocorrer através de aulas com instrutores ou *on job*. Os conhecimentos e as habilidades adquiridos através de treinamentos devem ser avaliados para verificar a devida eficiência. Deve existir uma base de dados contendo informações das habilidades, conhecimentos e treinamento já realizados pelos envolvidos com o projeto. Os treinamentos necessários para o projeto devem ser registrados no plano do projeto.

Na prática específica "*Planejar o envolvimento dos stakeholders*", é esperado que seja identificado no plano o envolvimento dos *stakeholders* no projeto. Os *stakeholders* devem ser identificados para todas as fases do ciclo de vida do projeto apontando a pessoa e sua função no projeto com a descrição do grau de relevância e interação nas atividades do projeto. Devem ser identificados os *stakeholders* que tenham a *expertise* necessária para conduzir as atividades do projeto de acordo com as fases do ciclo de vida do projeto.

Na prática específica "*Estabelecer o Plano de Projeto*", é esperado que todo o conteúdo do plano de projeto seja estabelecido e mantido, além de tratadas todas as questões do planejamento. É necessário que as informações contidas neste plano sejam de comum entendimento entre todos os envolvidos no projeto e que os compromissos assumidos sejam conhecidos e executados de acordo com o que foi planejado.

Os diversos planos que dão suporte ao plano de projeto devem estar com ele alinhados. O plano gerado para os projetos deve conter as seguintes informações: questões relacionadas a esforço, ciclo de vida do projeto definido, as atividades técnicas e gerenciáveis, custos e prazos, marcos do projeto, execução do gerenciamento dos dados do projeto, identificação dos riscos, necessidades de recursos e habilidades, identificação e a interação entre os *stakeholders*. A questão da descrição de infra-estrutura deverá ser contemplada incluindo as responsabilidades e as funções desempenhadas dentro da equipe de projeto, além de ser indicado como deve ser feito o acompanhamento das atividades com a gerência de alto nível e os membros pertencentes às equipes de suporte ao projeto.

Obter comprometimento com o plano

O objetivo específico "*Obter comprometimento com o plano*" exige que compromissos assumidos para o projeto sejam estabelecidos e mantidos. Para que o planejamento do projeto seja efetivo, deverá existir um comprometimento por todos os responsáveis pela implementação e pelo suporte ao plano.

Na prática específica "*Revisar planos que afetam o projeto*", é esperado que todos os planos que dão suporte ao plano de projeto, por exemplo, plano de gestão de configuração, plano da garantia da qualidade, plano de testes, etc., estejam alinhados, ou seja, as informações devem ser similares, para que os compromissos assumidos para o projeto sejam cumpridos. Assim, se todos os planos estiverem atualizados e refletindo as mesmas informações é possível que todos os envolvidos tenham conhecimento das mesmas informações.

A revisão de todos os planos que são afetados com alguma alteração realizada no plano de projeto é necessária, pois garante um entendimento comum do escopo, dos objetivos, dos papéis e da interação requeridas para o sucesso esperado do projeto.

Na prática específica "*Associar recursos e tarefas*", é esperado que o plano de projeto seja ajustado para refletir a disponibilidade real dos recursos em relação aos estimados. É preciso que sejam renegociados os recursos, caso as estimativas realizadas tenham sido subestimadas. Assim, obter o comprometimento dos *stakeholders* relevantes é importante para ajustar qualquer diferença existente entre os recursos estimados e os disponíveis.

Na prática específica "*Obter o compromisso com o plano*", espera-se que seja obtido o comprometimento dos *stakeholders* envolvidos responsáveis pela execução e suporte ao plano. Obter o comprometimento envolve a interação entre todos os *stakeholders* relevantes

tanto internos como externos ao projeto. Os compromissos assumidos devem assegurar que sua realização estará dentro dos custos, prazos e restrições de execução.

Somente planejar o projeto não é suficiente. É preciso garantir que, ao longo de sua execução, os planos estejam sendo cumpridos, mantendo-se a visibilidade sobre o progresso real do projeto em relação ao planejado, permitindo-se a tomada de ações corretivas, quando houver desvios do que foi inicialmente planejado. Assim, o passo seguinte é controlar o andamento das atividades e dos produtos do projeto a fim de garantir o cumprimento dos compromissos assumidos com os *stakeholders* relevantes. A seção seguinte apresentará a área de processo referente ao monitoramento e controle das atividades do projeto.

3.5 Monitoramento e Controle de Projeto

No nível 2 do CMMI, tem-se a área de processo "Monitoramento e Controle do Projeto", cujo propósito é prover o entendimento do progresso do projeto, de tal forma que ações corretivas apropriadas possam ser tomadas, quando a execução do projeto desvia-se significativamente de seu plano (SEI, 2001).

O plano do projeto documentado é a base para as atividades de monitoração, acompanhamento do status do progresso do projeto e tomadas de ações corretivas quando o desempenho do projeto de software desvia significativamente de seus planos. O progresso é determinado pela comparação entre os valores reais com os planejados. Estes valores a serem comparados são: atributos dos produtos de trabalho e tarefas (tamanho e complexidade), esforço, custo e prazo, além da verificação dos marcos e pontos de controle de acordo com o cronograma do projeto ou WBS. Uma visão adequada do projeto permite que uma ação oportuna seja tomada quando o desempenho do projeto desvie significativamente do planejado. Um desvio é considerado significante quando, deixado de ser resolvido, impossibilita o andamento do projeto levando ao não alcance dos objetivos e compromissos assumidos.

Quando o status real do projeto desvia significantemente dos valores previstos, ações corretivas são tomadas apropriadamente. Estas ações podem requerer um replanejamento do projeto, como uma revisão do plano, estabelecimento de novos acordos ou inclusão de atividades adicionais de mitigação no plano de projeto.

Quando ocorrem alterações nos planos por conta de ações corretivas, é fundamental que os planos originais sejam mantidos. Isto permitirá a *posteriori* que erros de estimativas sejam conhecidos e, portanto, reduzidos. Localizando-se o erro, pela comparação entre o planejado e o realizado, futuros projetos poderão ser estimados com maior precisão. Quando

isto não é feito, pode-se ter a impressão de que o projeto está sempre em dia, já que os planos foram refeitos.

Os objetivos específicos e as práticas específicas relacionadas a cada objetivo específico são os seguintes:

- Monitorar o projeto de acordo com o plano:
  - Monitorar os parâmetros do planejamento de projeto;
  - Monitorar os compromissos;
  - Monitorar os riscos do projeto;
  - Monitorar o gerenciamento dos dados;
  - Monitorar o envolvimento dos *stakeholders*;
  - Conduzir revisões do progresso;
  - Conduzir revisões de marcos.
- Gerenciar as ações corretivas até sua conclusão:
  - Analisar problemas;
  - Tomar ação corretiva;
  - Gerenciar ação corretiva.

A seguir cada prática específica apresentada anteriormente será detalhada.

Monitorar o projeto de acordo com o plano

O objetivo específico "*Monitorar o projeto de acordo com o plano*" descreve que algum desvio identificado durante a execução do projeto em relação ao planejado deve ser monitorado. O progresso do projeto deve ser acompanhado com base nos dados de planejamento. Caso os dados reais estejam divergindo muito do que havia sido estimado, ações corretivas devem ser tomadas.

Na prática específica "*Monitorar os parâmetros do planejamento de projeto*", é esperado que os parâmetros de planejamento de projeto sejam acompanhados no decorrer da execução do projeto. Os parâmetros de planejamento de projeto constituem os indicadores típicos que mostram o progresso do projeto. São eles: atributos de produto de trabalho e tarefas, custo, esforço e prazo. Os atributos dos produtos de trabalho e tarefas podem ser itens do tipo: complexidade, tamanho, peso, método, adaptação do projeto, funções a serem desenvolvidas.

Monitorar envolve tipicamente analisar os valores atuais do projeto com base nos parâmetros do planejamento do projeto, comparar os valores reais com o estimado no plano e identificar os desvios significativos encontrados no projeto.

Deve-se monitorar: o cronograma do projeto, verificando se o projeto está ou não atrasado em relação ao planejado; se os principais marcos foram cumpridos conforme estabelecido no plano de projeto; se o custo, o esforço despendido no projeto e os atributos dos produtos de trabalho e tarefas estão de acordo com o estimado; e se os recursos planejados para o projeto foram fornecidos adequadamente. Os desvios apresentados devem ser documentados.

Na prática específica "*Monitorar os compromissos*", é esperado que os compromissos (internos e externos) assumidos com o projeto sejam satisfeitos dentro do planejado. Devem ser identificados os compromissos que não foram satisfeitos e os riscos significativos, caso um compromisso não tenha sido cumprido. Revisões regulares dos compromissos assumidos para o projeto devem ser realizadas.

Na prática específica "*Monitorar os riscos do projeto*", é esperado que os riscos identificados no plano de projeto sejam monitorados. Periodicamente, devem-se revisar os riscos do projeto e documentar as revisões dos riscos. O *status* dos riscos deve ser registrado e informado aos *stakeholders* relevantes.

Na prática específica "*Monitorar o gerenciamento dos dados*", é esperado que seja monitorado o gerenciamento dos dados do projeto planejados. No plano de projeto, foram estabelecidos os tipos de dados que o projeto gostaria que fossem gerenciados. Nesta área de processo, o gerenciamento destes dados será acompanhado com base no planejado. Revisões periódicas serão realizadas para verificar a eficiência do gerenciamento dos dados para o projeto. Os problemas e impactos identificados nestas revisões devem ser registrados.

Na prática específica "*Monitorar envolvimento dos stakeholders*", é esperado que o envolvimento dos *stakeholders* seja executado de acordo com o planejamento do projeto. O envolvimento dos *stakeholders* deve ser monitorado para garantir que interações apropriadas estejam acontecendo conforme o planejado. Periodicamente, deve-se revisar o status dos *stakeholders* envolvidos, identificar e documentar os problemas significativos e seus impactos e documentar os resultados das revisões dos status do envolvimento dos *stakeholders*.

Na prática específica "*Conduzir revisões de progresso*", é esperado que, periodicamente, o progresso, a execução das atividades e os problemas do projeto sejam revistos. As revisões do progresso são realizadas no projeto para manter os *stakeholders* informados sobre o andamento e status do projeto. Estas revisões podem ser informais e ser realizadas sem que tenham sido previamente planejadas.

Nesta prática específica, os seguintes pontos são tratados: (*i*) acompanhamento do status das atividades e produtos de trabalho dos *stakeholders* relevantes; (*ii*) revisão dos

resultados das coletas e análises das medições para o controle do projeto; (*iii*) identificação e documentação dos problemas e desvios significativos em relação ao plano, documentação das solicitações de mudanças, e identificação de problemas em qualquer produto e processo de trabalho (com relação ao gerenciamento de configuração); (*iv*) documentação dos resultados das revisões; e (*v*) acompanhamento das solicitações de mudanças e problemas até seu fechamento.

Com relação à prática específica, este trabalho propõe uma forma para realizar o acompanhamento do *status* do progresso funcional (capítulo 5), ou seja, acompanhar o *status* do tamanho do projeto baseado na complexidade dos casos de uso.

Um ponto crítico dentro da gerência de projetos é a falta de técnicas de acompanhamento do progresso, que permitam visualizar o *status* do projeto de modo mais técnico, apresentando claramente o que foi desenvolvido e o que falta desenvolver (Meneses, 2001).

Na prática específica "*Conduzir revisões de marcos*", objetiva-se que sejam revistos o cumprimento dos acordos e resultados esperados pelo projeto nos marcos preestabelecidos.

As revisões em marcos são planejadas e são tipicamente revisões formais. Nesta prática específica, revisões são conduzidas em pontos importantes do cronograma do projeto, tal como finais de fases do projeto. No planejamento do projeto, são indicados os envolvidos para cada tipo de revisões para cada marco. São revistos compromissos, planos, status do projeto e os riscos. Os problemas e seus impactos apresentados nas revisões são identificados e documentados. Os resultados das revisões, ações e decisões tomadas diante dos problemas são documentados e as ações geradas são acompanhadas até o seu fechamento.

Gerenciar as ações corretivas até sua conclusão

O objetivo específico "*Gerenciar ações corretivas até sua conclusão*" descreve que ações corretivas devem ser tomadas e acompanhadas até o seu fechamento, quando são apresentados resultados do projeto com desvios significantes do planejado.

Na prática específica "*Analisar problema*", é esperado que sejam coletados e analisados os problemas e que ações corretivas sejam determinadas, para tratar estes problemas de maneira adequada. Os problemas são coletados e listados para análise. Ações corretivas são necessárias para cada problema indicado.

Na prática específica "*Tomar ação corretiva*", é esperado que para os problemas identificados sejam realizadas ações corretivas. Um planejamento para a execução de ações deve ser elaborado e os *stakeholders* relevantes, que estejam envolvidos diretamente com a

realização desta atividade, devem assumir este novo compromisso com o projeto. Pode ser que alguns compromissos assumidos pelo projeto sejam afetados por causa das ações corretivas a serem realizadas; assim, uma possível negociação de compromissos internos ou externos é necessária.

Na prática específica "*Gerenciar ação corretiva*" é esperado que sejam gerenciadas as ações corretivas até a sua conclusão. As ações corretivas são monitoradas até que seja totalmente finalizada. No entanto, a efetividade da ação corretiva é obtida através da análise dos resultados da ação. As ações corretivas adequadas para corrigir desvios com relação ao planejado devem ser documentadas para que sejam utilizadas como lições aprendidas para projetos posteriores.

3.6 Conclusão

Neste capítulo, foram apresentados modelos de processos de software CMM e CMMI e a norma ISO 15504. Foi detalhada a questão do planejamento e acompanhamento de projeto através das áreas de processo do CMMI: *Planejamento de Projeto* e *Monitoração e Controle de Projeto*, que está relacionada com a elaboração de estimativas, foco deste trabalho.

No capítulo seguinte, serão apresentados os tipos de métricas e algumas técnicas de estimativas utilizadas no planejamento de sistemas de software.

Capítulo **4**

Estimativas de Projeto de Software

Este capítulo apresenta conceitos relevantes relacionados à medição e às métricas de estimativa de projeto de software, além de mostrar algumas técnicas importantes de estimativas.

Um dos principais riscos que atinge o processo de estimativas é a falta de credibilidade que lhes conferem as equipes de desenvolvimento (BOEHM, 2000). Isto ocorre quando as estimativas irreais, compõem projetos freqüentemente subestimados ou superestimados. Neste contexto, a precisão das estimativas de tamanho torna-se fundamental para a elaboração de cronograma e orçamento realistas, por constituírem a base para a derivação das estimativas de esforço, prazo e custo (SEI, 2002).

As estimativas apóiam essencialmente as atividades de gestão de projetos de software. Estimativas eficientes permitem a verificação da viabilidade do projeto, a elaboração de propostas técnicas e comerciais, a confecção de planos e cronogramas detalhados e o acompanhamento efetivo de projetos.

Eficiência, entrega do produto no prazo, dentro do orçamento previsto e com um nível de qualidade desejado pelo cliente são características que influenciam atualmente o sucesso de organizações de desenvolvimento de software no mercado globalizado e competitivo da Tecnologia da Informação (TI). Assim sendo, ressaltam-se a importância de mecanismos de acompanhamento e a avaliação do progresso do processo, projeto e produto.

Esses mecanismos, normalmente baseados em informações qualitativas e quantitativas, coletadas através de medições realizadas durante o projeto, são recursos essenciais na gestão do desenvolvimento de software. Para o gerente do projeto, essas medidas, também conhecidas como métricas, permitem um melhor entendimento do processo de produção e do controle do projeto de software. Desta maneira, as medições fornecem informações que permitem a determinação de pontos fortes e fracos dos processos e produto, indicam ações corretivas e propiciam avaliações de impacto de tais ações. Enfim, garantem a

qualidade do processo e dos produtos obtidos (FENTON; PFLEEGER, 1997; BASILI; CALDIERA; ROMBACH, 1994).

A necessidade de medidas que retratem a eficiência do desenvolvimento e minimizem os fracassos de projetos, principalmente em relação a falhas no cronograma e em estimativas realizadas, demandaram a realização de estudos na área de estimativas de tamanho de software, que resultaram na proposição de diversas métricas e métodos de medição de projeto, processo e produtos de software.

4.1 Métricas de Software

A medição é fundamental para qualquer atividade de engenharia; a engenharia de software não é a exceção. Na engenharia civil, por exemplo, é possível que um engenheiro possa planejar a construção de uma ponte com precisão confiável de que o valor estimado para custo e tempo seja próximo do real. Muitas vezes, na engenharia de software, o planejamento de um projeto pode ter uma variação nas estimativas devido à falta de dados históricos, e à disponibilização de reduzido número de técnicas maduras (PRESSMAN, 2002).

Podem-se elencar algumas das razões que tornam o planejamento e a estimativa de um projeto de software tão ineficientes, ao contrário de outras engenharias:

- o software é único.
- nunca se tem o mesmo projeto ou produto de software duas vezes.
- o desenvolvimento de software é uma disciplina baseada no conhecimento que freqüentemente requer inovação e experimentação. Como planejar isso?;
- a tecnologia da informação muda muito rapidamente, e algumas métricas baseadas em um conjunto de tecnologia podem não ser mais aplicáveis a um outro determinado conjunto (POLLICE, 2004).

4.1.1 Medida, Medição, Métrica e Indicadores

Uma medida é a indicação quantitativa de um atributo, como o tamanho, a dimensão, a capacidade de um produto ou de um processo. (PRESSMAN, 2002). Um exemplo de medida, que é um número bruto (POLLICE, 2004), é o número de defeitos encontrado em um programa.

A medição é o ato de determinar uma medida (POLLICE, 2004), de se medir o número de defeitos no código.

A métrica é uma indicação quantitativa de uma medida para que um sistema ou componente possua um atributo (PRESSMAN, 2002). Assim, uma métrica poderia ser, por exemplo, a quantidade de defeitos por centena de linhas de código. Em outras palavras, a métrica é descrita em termos de relacionamento entre as medidas (POLLICE, 2004).

O *IEEE Standard Glossary of Software Engeneering Terms* (IEEE, 1993) define métrica como "medida quantitativa do grau de um sistema, componente ou processo que possui determinado atributo". Quando os dados de um único ponto são coletados, como, por exemplo, a quantidade de erros descobertos em uma revisão de um módulo, uma medida é estabelecida. A medição ocorre como resultado da coleta de um ou mais pontos.

Um indicador é uma métrica, ou uma combinação de métricas, que fornece a compreensão de um projeto, processo ou produto de software propriamente dito (RAGLAND, 1995). Um indicador fornece a compreensão que possibilita ao gerente de projetos ou aos engenheiros de software ajustar adequadamente o projeto, processo ou produto.

4.1.2 Classificação das Métricas

As métricas podem ser classificadas em várias categorias: métricas de processo, produto e recursos (FENTON; PFLEEGER, 1997), objetivas e subjetivas (MÖLLER, 1993; PAULISH, 1993) e diretas e indiretas (FENTON; PFLEEGER, 1991; PRESSMAN, 2002).

As métricas de processo medem as atividades realizadas durante todo o desenvolvimento de software; as métricas de produto medem os artefatos, produtos e documentos gerados pelo processo; e as métricas de recursos medem as entidades requeridas para a execução do processo (FENTON; PFLEEGER, 1997).

As métricas objetivas podem ser quantificadas e medidas através de expressões numéricas ou representações gráficas de expressões numéricas contadas a partir do código fonte, projeto, dados de teste e outras informações do software. As métricas subjetivas são medidas baseadas em estimativas pessoais ou de grupo, geralmente obtidas através de conceitos como excelente, bom, regular e ruim (MÖLLER, 1993; PAULISH, 1993).

Segundo Fenton e Pfleeger (1997), métricas subjetivas dependem da pessoa que está mensurando, do seu julgamento e do grau de imprecisão, o que dificulta o consenso entre possíveis atributos que envolvam processos, produtos ou qualidade.

As métricas diretas são aquelas que não dependem da medida de outro atributo, mas da quantificação de um fator observado no produto. Já as métricas indiretas envolvem as medidas de um ou mais atributos a estes relacionados (FENTON; PFLEEGER, 1997).

Segundo Pressman (2002) as medidas diretas são tudo aquilo que se pode medir com maior precisão. Por exemplo, comprimento de um parafuso, o custo e o esforço aplicado no desenvolvimento e na manutenção do software e do produto, linhas de código e velocidade de execução. Já as medidas indiretas são difíceis de ser avaliadas, pois não se consegue facilmente medir, por exemplo, qualidade, eficiência, funcionalidade, complexidade, etc.

Segundo McGarry *et al*. (2001), destacam-se algumas abordagens de métricas de estimativas de projeto de software dentro da categoria das métricas diretas, como: (*i*) analogia; (*ii*) julgamento de especialistas (*iii*) modelos paramétricos; (*iv*) atividades baseadas em modelos; e (*v*) relacionamentos de estimativas. As abordagens mais comuns são por analogia, julgamento de especialistas e por modelos paramétricos.

A analogia é geralmente realizada por especialista em estimativas (com base na experiência de projetos anteriores) ou por algoritmos de busca de projetos similares (disponíveis em base de dados) (JORGENSEN; INDAHL; SJOBERG, 2003). Esta abordagem requer um conhecimento detalhado do projeto, para identificar as diferenças específicas entre o projeto proposto e projetos anteriores, que foram usados como base para realizar a estimativa. Segundo Heemstra (1992 apud JORGENSEN; INDAHL; SJOBERG, 2003), a estimativa por analogia é o método mais comum de estimativa na indústria de software.

O julgamento de especialistas é realizado com base nas experiências de profissionais em projetos semelhantes. As técnicas baseadas na experiência de especialistas são úteis na ausência de dados quantitativos. Longstreet (2002) sugere que ao "avaliar projetos semelhantes, deve-se considerar o tipo de plataforma de hardware, de linguagem, de projeto, de sistema operacional e identificar os dados históricos de taxa de entrega, de cronogramas e de custo do projeto".

Os modelos paramétricos assumem que existe um relacionamento matemático entre o tamanho, esforço, prazo e qualidade e que o relacionamento é afetado por fatores mensuráveis de desempenho também chamados parâmetros. A entrada mais importante nesse modelo é o tamanho, o qual representa a quantidade de funções do software. Para obter melhores resultados, os modelos paramétricos devem ser calibrados com dados do ambiente local de desenvolvimento (McGARRY *et al*., 2001).

Todas as abordagens de métricas de estimativas requerem alguns dados históricos similares ao projeto proposto, incluindo o domínio da aplicação, o nível de maturidade do processo de desenvolvimento, a experiência da equipe, taxas de produtividade, tipos de tecnologia e de ferramentas utilizadas e o grau de reúso para se obter resultados significativos (FENTON; PFLEEGER, 1997; McGARRY *et al*., 2001).

Segundo Fenton e Pfleeger (1997), para as principais métricas diretas (analogia, julgamento de especialistas e modelos paramétricos), duas abordagens de estimativas ainda podem ser usadas: *top-down* e *bottom-up*.

As estimativas *top-down* (GARMUS; HERRON, 2000; FENTON; PFLEEGER, 1997; KARNER, 1993; McPHEE, 1999; RIBU, 2001) são utilizadas para estimar o projeto, quando informações sobre o escopo do projeto são ainda muito limitadas, pois estimam o projeto como um produto único. Os componentes, as atividades e as fases do projeto são calculados como porções relativas ao todo estimado. A granularidade das estimativas dos produtos de trabalho é determinada com base na estimativa total do projeto (MYLIUS; MARODIN, 2004).

As estimativas *top-down* têm a vantagem de exigir muito menos esforço para elaboração e pouco tempo para serem desenvolvidas, mas com a desvantagem é sua precisão pouco confiável.

As estimativas *bottom-up* (BOEHM, 1981) são obtidas primeiramente através das estimativas das partes do projeto e a estimativa total do projeto é obtida por meio da soma de suas partes. Essa abordagem envolve estimar a atividade ou produto a ser executado no projeto, depois sumarizá-lo ou agregá-lo para obter uma estimativa total do projeto.

Referida abordagem solicita que primeiramente sejam levantadas todas as atividades para o projeto, a partir da elaboração de uma WBS[1], contendo todas as possíveis atividades a serem realizadas no projeto.

A vantagem desse tipo de estimativa é que ela gera resultados precisos. Sua precisão depende do nível de detalhe que está sendo considerado. Sua desvantagem é a ausência de alguma importante a ser contemplada na WBS. Isso reflete a dificuldade na estimativa do esforço requerido para as atividades de *overhead* que não são incluídas na WBS, mas que são claramente necessárias (JALOTE, 2001). Além disso, esse tipo de abordagem exige um esforço considerável e um custo com relação ao detalhamento de todas as possíveis atividades do projeto.

Segundo Boehm (1998), a abordagem *bottom-up* é complementar à abordagem *top-down*. A precisão das estimativas também depende de um detalhamento maior das informações do projeto e de uma base histórica de projetos semelhantes já concluídos.

Ambas as abordagens de estimativas descritas anteriormente podem utilizar dados já documentados de projetos anteriores similares, para fornecer suporte à elaboração de novas

[1] WBS (*Work Breakdown Structure*) ou Estrutura Analítica de Trabalho é um agrupamento dos elementos de um projeto, que organiza e define o escopo total do trabalho (DINSMORE; CAMPBELL, 2003).

estimativas de projetos. Para isto, os seguintes fatores devem ser considerados, por influenciarem a estimativa do esforço do projeto: o mesmo ciclo de vida do projeto, a mesma metodologia de desenvolvimento, as mesmas ferramentas, e o uso de uma equipe de projeto com habilidades e experiências similares.

4.2 Processo de Estimativa

Segundo Vasquez (2003), o processo de estimativa de um projeto de software envolve quatro atividades básicas (Figura 4.1):

- estimar o tamanho do produto a ser desenvolvido;
- estimar o esforço empregado na execução do projeto;
- estimar o prazo (duração) do projeto;
- estimar o custo do projeto.

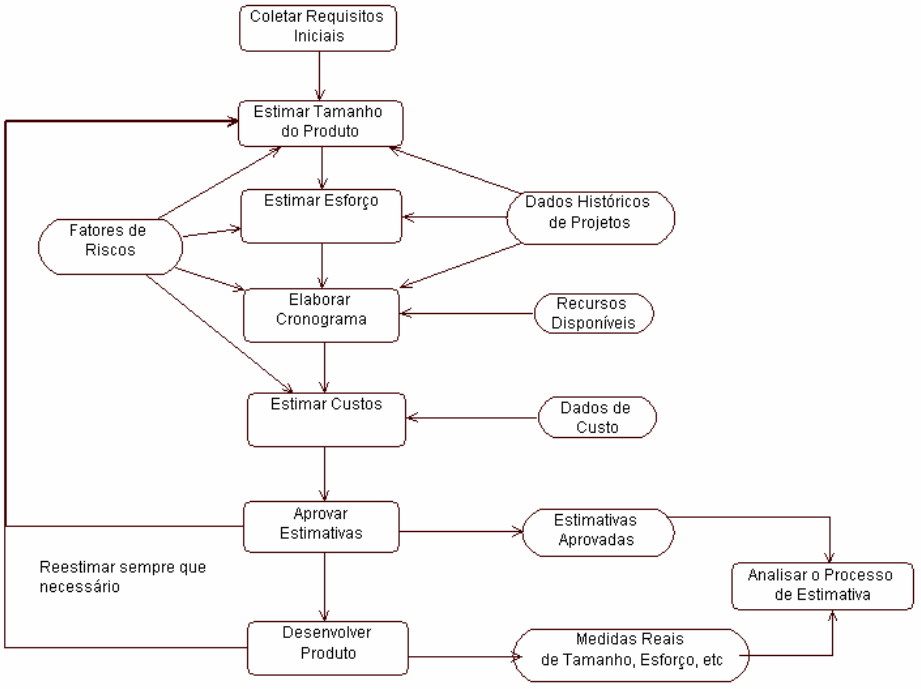


Figura 4.1 – Processo de estimativa de um projeto de software (VASQUEZ, 2003)

O propósito principal de um processo de estimativa é fornecer informações que beneficiem o planejamento e o controle dos projetos de software o mais cedo possível durante seu ciclo de vida (VASQUEZ, 2003).

O processo de estimativa inicia-se a partir dos primeiros níveis de abstração dos requisitos do projeto. Para ser completo e eficiente, deve levar em consideração as

experiências e os dados históricos de projetos passados, os recursos disponíveis dentro e fora da organização, os dados de custo e os fatores de riscos que envolvem os projetos.

A cada etapa realizada, as estimativas obtidas devem passar por um processo de aprovação, antes de serem registradas na base histórica. Além de poderem ser utilizadas como fonte de informação para projetos futuros, servirão como insumos para as estimativas das etapas seguintes.

Reestimar deve ser uma atividade constante durante todo o ciclo de vida do desenvolvimento do software. Na etapa final do processo, após a conclusão do produto, as medidas reais de tamanho, esforço, duração e custo devem ser registradas na base histórica, servindo também para a validação e o aprimoramento de todo o processo de estimativa.

O primeiro passo para se obter estimativas efetivas do projeto de software é estimar o seu tamanho. A partir daí, as estimativas de esforço, prazo e custos poderão ser determinadas.

A realização de estimativas é essencial no que diz respeito às questões de negócio de uma empresa. Além disso, é o suporte para o planejamento e o gerenciamento de um projeto. Na elaboração de uma proposta de negócio entre cliente e fornecedor, as estimativas de custo e prazo devem ser conhecidas, para que se faça a análise de uma proposta viável ou não para ambos. Freqüentemente, nos contratos firmados são incorporadas e fixadas estimativas de custo e prazo. Como resultado, o fornecedor precisa ter estimativas precisas, pois um projeto subestimado poderá prejudicá-lo, já que o cliente pagará apenas o custo estipulado no contrato assinado e não sobre valores não previstos depois do fechamento do contrato.

A seguir, serão apresentadas as estimativas básicas para o processo de software.

4.2.1 Estimativas de Tamanho

O tamanho do software significa a quantidade de trabalho a ser executado no desenvolvimento de um projeto em uma unidade de medida especificada. Cada projeto pode ser estimado de acordo com seu tamanho físico (que pode ser medido através de especificação de requisitos, análise, projeto e código), com base nas funções que o usuário obtém, na complexidade do problema que o software irá resolver e no reúso do projeto (que mede o quanto o produto será copiado ou modificado a partir de um outro produto existente) (FENTON; PFLEEGER, 1997; McPHEE, 1999).

Estimar tamanho do produto de software é um dos primeiros passos para se efetivar uma estimativa. As fontes de informação com relação ao escopo do projeto iniciam-se na fase de concepção com a descrição formal de requisitos. Por exemplo, uma solicitação de proposta do cliente, uma especificação de requisitos do sistema, ou uma especificação de casos de uso.

Quanto maior for o conhecimento e o detalhamento das informações sobre o projeto melhor serão os subsídios para a elaboração da estimativa de tamanho.

As estimativas geralmente acontecem em fases iniciais de um projeto. Neste momento, o entendimento sobre o que deve ser desenvolvido ainda é incipiente. Em fases posteriores, a estimativa deve ser refeita quando mais informações sobre o projeto tornarem-se disponíveis (JALOTE, 2001).

Há autores que destacam que a estimativa de tamanho é uma das atividades mais difíceis de ser realizada em uma organização de desenvolvimento de software, principalmente quando é realizada no início do projeto (ROSS, 2004).

Atualmente, existem diversas métricas de estimativa de tamanho de projetos de software. Entretanto, não é trivial a seleção mais apropriada de uma dessas métricas para um organização. As métricas mais conhecidas e utilizadas atualmente foram desenvolvidas com base nas funções de software tais como: *Function Point Analysis* (Análise por Pontos de Função); *Bang*; *Features Points*; *Boeing´s ED Function Point Analysis*, MK II *Function Point Analysis*, COSMIC *Full Function Point*, *Internet Points*, *Domino Points*, *Use Case Points* (Pontos de Caso de Uso), *Class-Method Points* (GARMUS; HERRON, 2000; McPHEE, 1999; ROETZHEIM, 2000). Algumas dessas métricas serão apresentadas no decorrer deste capítulo.

4.2.2 Estimativas de Esforço

O esforço estimado para o projeto pode ser obtido a partir do cálculo da estimativa de tamanho do produto. O esforço total de um projeto é calculado com base em seu processo de desenvolvimento. Esse processo envolve muito mais do que a simples atividade de codificação do software – o fato é que a codificação é freqüentemente a menor parte do esforço real de todo o projeto. Elaboração de documentos, implementação de protótipos, projeto do produto a ser entregue, revisão e teste do código levam uma grande fatia de todo o esforço do projeto (PETERS, 1999).

A melhor maneira de se chegar à estimativa de esforço a partir da estimativa de tamanho do projeto é utilizar-se de dados internos da própria organização, como o número de horas alocadas em projetos passados, conjuntamente com o tamanho do projeto, dimensionado em suas diversas fases (VASQUEZ, 2003).

Embora as estimativas de tamanho, esforço e prazo sejam necessárias para o planejamento, a estimativa de prazo torna-se mais fácil de ser definida se a estimativa de esforço já for conhecida. Daí, o foco nos projetos de software comumente definir a estimativa

do esforço, embora algumas estimativas sejam dependentes da estimativa de tamanho de um produto.

A falta de estimativas de esforço e do tempo necessário para executar um projeto, podem tornar inviável o planejamento e o gerenciamento de um projeto de software. Um requisito fundamental para um planejamento efetivo é definir as estimativas de esforço e de prazo para o projeto e para suas várias atividades. Sem essas estimativas não se pode saber se o projeto está atrasado ou se seu custo foi maior do que o estimado (JALOTE, 2001).

O esforço tem sido estimado pelo *feeling* ou por experiências anteriores de gerentes de projetos em organizações consideradas imaturas em seu processo de software, o que nem sempre tem dado resultados efetivos.

4.2.3 Estimativas de Prazo

Com as estimativas de tamanho e esforço calculadas para um projeto de software podem-se, então, determinar as estimativas de prazo. Isto geralmente demanda estimativa de quantas pessoas estarão envolvidas no projeto, que atividades (WBS) serão executadas, e quando essas atividades iniciarão e serão finalizadas. Tanto dados históricos quanto modelos de dados da indústria podem ser usados para predizer o número de pessoas necessárias para um tamanho de projeto e como o trabalho pode ser dividido dentro do prazo previsto.

As atividades de estimativas de prazo em projetos de software costumam ser assistidas por ferramentas de planejamento de projeto que contenham cronograma.

Segundo McConnell (1996) e Peters (1999), a estimativa de prazo pode ser obtida através da equação *Eq*. 4.1.

$$\textbf{\textit{Prazo}}\ (\textbf{\textit{meses}}) = 3{,}0 * \textbf{\textit{Esforço}}\ (\textbf{\textit{meses}})^{1/3} \qquad (Eq.\ 4.1)$$

As opiniões variam em relação ao fator numérico: 2,0 ou 2,5 ou até mesmo 4,0 devem ser usados no lugar de 3,0. A calibração deste valor deve ser definida somente através de tentativas em relação às características da organização,.

No planejamento do desenvolvimento ou da manutenção de um sistema, a estimativa de sua duração é uma das atividades que exigem maior experiência e conhecimento dos aspetos envolvidos em sua execução. Pela utilização intensa de trabalho humano, é principalmente avaliada pela relação da estimativa de trabalho envolvido e os recursos disponíveis para sua execução.

A estimativa de prazo de maneira simplificada pode ser dada também pela razão entre o esforço previsto (geralmente em número de horas trabalhadas) e a quantidade de recursos alocada na execução do projeto, conforme *Eq.* 4.2.

$$\textbf{\textit{Prazo}} = \frac{\textbf{Esforço}}{\textbf{Quantidade de Recursos}} \qquad (Eq.\ 4.2)$$

Estimativas de prazo e custo devem ser elaboradas para todos os projetos de software, pois é objetivo básico na execução de um projeto realizá-lo dentro de prazos e custos estimados conforme o contrato. O grau de formalismo para se determinar essas estimativas pode diferir de um projeto para outro, dependendo de suas necessidades, de métodos prognósticos e da disponibilidade de ferramentas.

4.2.4 Estimativas de Custo

Em projetos de software, o custo é comumente proporcional ao esforço despendido para sua construção, onde o trabalho humano é o principal recurso a ser consumido. Conseqüentemente, o custo é com freqüência associado a homens-mês ou homens-hora.

A estimativa de esforço é convertida para o custo real usando-se uma proporção padrão de esforço por unidade. Essa proporção leva em conta outros custos, como o custo de hardware e de infra-estrutura. Desta maneira, as estimativas não somente ajudam a definir o custo, mas também auxiliam no planejamento das necessidades dos recursos humanos do projeto.

Em relação às estimativas de custos, existem muitos fatores a ser considerados, quando se está estimando o custo total de um projeto de software, o que pode incluir horas trabalhadas, aquisição ou aluguel de hardware e software, viagens com reuniões e testes no cliente, contas telefônicas (ligações de longas distâncias, vídeo-conferência, linhas dedicadas para testes, etc.), treinamentos, entre outros custos.

Estimar o custo total do projeto depende em geral de como a organização os aloca. Alguns custos podem não ser alocados por projeto, mas ser considerados como valores de *overhead* [2] de horas trabalhadas. Freqüentemente, um gerente de projeto de software estima somente o custo de horas trabalhadas e identifica alguns custos adicionais de projeto que não são considerados *overhead* para a organização.

O custo total das horas trabalhadas pode ser obtido pelo produto da estimativa de esforço do projeto (em horas) e valor de uma hora trabalhada (R$ por hora). Um custo total

[2] Despesas gerais, despesas indiretas.

mais preciso pode ser obtido através do valor das horas trabalhadas específicas de cada recurso utilizado no projeto (recursos técnicos, recursos de suporte, gerente de projeto, etc.). Uma outra maneira de se obter o custo total das horas trabalhadas é determinar o percentual do esforço total do projeto a ser realizado em cada etapa do processo pelos recursos do projeto.

Independentemente da etapa do processo, existem várias técnicas de estimativas que podem ser utilizadas, como apresentadas a seguir, segundo Arifoglu (1993), Bonfin (2001), Karner (1993), Pressman (2002), e Ribu (2001).

4.3 Linhas de Código – LOC

As primeiras métricas de estimativa de tamanho de software basearam-se no tamanho físico de linhas de código como o *Lines of Code* (LOC) (ARIFOGLU, 1993; FENTON; NEIL, 2000; KEMERER, 1992). Essa métrica considera o software sob a perspectiva de sua estrutura interna e é aplicada nas fases finais do projeto (MISIC; TESIC, 2004).

Como o próprio nome sugere, essa métrica propõe que o tamanho de um sistema deve ser medido pela contagem das linhas de seu código fonte. Como a quantidade exata de linhas de código só é conhecida após a conclusão do sistema, a estimativa pode ser feita de duas formas (SIMON, 2000):

- através de variáveis de estimativa usadas para cada elemento do sistema;
- com o uso de dados de projetos anteriores, juntamente com variáveis de estimativa, projetando-se o custo e esforço de novos projetos.

Percebe-se que a métrica LOC é muito frágil e imprecisa, tendo várias desvantagens:

- a definição de linha de código é obscura, sem padrões, não estando claro se deve incluir ou não declarações de dados, macro-instruções, etc. (SIMON, 2000);
- é uma medida técnica, sem significado para o usuário (DUMKE, 1999; ROSS, 2004);
- não é consistente, pois é, algumas linhas de código são mais trabalhosas que outras (DEMARCO, 1989);
- apresenta problemas de definição para linguagens não procedurais (PRESSMAN, 2000).

Além dessas desvantagens, Jones (1994) *apud* Fenton e Pfleeger (1997) ressaltam que uma contagem em LOC depende do grau de reúso e da linguagem de programação e pode ser até cinco vezes superior a uma outra estimativa, devido às diferenças das técnicas de medição de linhas em branco, linhas de comentário, declaração de dados e linhas de instruções. Esta

métrica penaliza programas pequenos e bem projetados, não se adapta às linguagens não procedimentais e é de difícil obtenção na fase inicial de planejamento.

Apesar de suas desvantagens, a métrica LOC não deve ser completamente descartada. As organizações podem usá-la para medir o que de fato foi produzido. Além disso, possui alguns pontos relevantes:

- simplicidade e farta literatura disponível (PRESSMAN, 2000);
- serve como medida de normalização para qualidade de software, como, por exemplo, quantidade de defeitos por linhas de código (FENTON, 1999);
- pode-se avaliar a produtividade (FENTON, 1999);
- a contabilização das linhas de código pode ser feita automaticamente;
- a possibilidade de se fazer estimativas automaticamente e a facilidade de uso de dados históricos (ROSS, 2004).

LOC foi uma métrica bastante utilizada até meados da década de setenta. A partir daí, surgiram diversas linguagens de programação e, conseqüentemente, a necessidade de se ter outras formas de estimar o tamanho de software.

Atualmente, são várias as métricas de estimativa de tamanho existentes e há dificuldades para se selecionar a mais apropriada para o tamanho de um projeto de software de uma organização. Muitas dessas métricas foram desenvolvidas com base nas seguintes funções de software: *Function Point Analysis* (Análise por Pontos de Função); *Bang*; *Feature Points*; *Boeing´s ED Function Point*; MK II *Function Point Analysis* (MK II FPA); *Use Case Points* (Pontos de Casos de Uso), *COSMIC Full Function Point* (GARMUS; HERRON, 2000; MCPHEE, 1999).

4.4 Análise de Pontos de Função – FPA

No princípio da década de setenta, pesquisadores da IBM iniciaram estudos para determinar que variáveis críticas poderiam determinar a produtividade na programação. Descobriram que um sistema seria mais bem avaliado através das funções executadas pelos programas, em vez de considerar o volume ou a complexidade do código.

Em 1979, Albrecht (1979) definiu a Análise de Pontos por Função (*Function Point Analysis* - FPA) a partir destes estudos, buscando mapear as questões pertinentes à estimativa e avaliação de produtividade no desenvolvimento de software em ambientes heterogêneos. Segundo Smith (1997), as primeiras definições da FPA foram refinadas e estendidas no IBM CIS *Guideline* 313, AD/M *Productivity Measurement and Estimate Validation*, em 1984. Em 1986, um grupo de usuários da FPA formou o *International Function Point User Group*

(IFPUG), o qual é responsável por manter seus associados informados a respeito das novas atualizações da técnica.

A FPA é uma das primeiras métricas a medir o tamanho do software com alguma precisão, tornando-se a métrica mais usada e estudada na história da engenharia de software. No final de 1993, já havia grupos de usuários em 18 países (JONES, 1995).

A FPA tornou-se um padrão internacional em 2002 através da norma ISO/IEC 20.926 (DEKKERS, 2003). Além disto, o mapeamento da FPA para estimativas de projetos de software OO tem sido largamente discutido na literatura.

Segundo Pressman (2000), a FPA é uma métrica orientada à função, derivada a partir de medidas diretas, que dimensionam o software considerando a funcionalidade entregue ao usuário final. Em síntese, o projeto lógico deve ser inteiramente decomposto em funções de acordo com os arquivos de dados, interfaces, entradas, saídas e consultas, atribuindo-se pesos a cada uma dessas funções. Esses pesos são multiplicados pelas quantidades de cada função e posteriormente somados. Finalmente, o somatório é ajustado conforme uma análise das características gerais de complexidade do sistema.

A FPA visa estabelecer uma medida de "tamanho" do software, em Pontos por Função (PF) (unidade de medida para software assim como a hora é unidade de medida para tempo), através da quantificação das funções implementadas sob o ponto de vista do usuário (LONGSTREET, 2002).

Conforme Tavares (1999), Ho (1999) e IFPUG (1999) os objetivos da FPA são:

- medir o que foi requisitado e recebido pelo usuário;
- prover uma métrica para a qualidade e análise de produtividade;
- fornecer um método para estimar o tamanho do software, independentemente da tecnologia utilizada para implementação e sob a perspectiva do usuário; e
- prover um fator de normalização para comparação de software.

O IFPUG (1999) tem evidenciado duas preocupações fundamentais em relação ao processo de contagem dos pontos por função:

- manter sua simplicidade para evitar acréscimo excessivo de trabalho; e
- sustentar sua concisão para garantir sua consistência, ao longo do tempo, dos projetos e entre os usuários da técnica.

Embora a idéia original tenha sido prover uma forma para medir tamanho de projetos de software, a FPA tem sido utilizada em outras atividades, tais como (IFPUG, 1999; HELLER, 1995; TAVARES, 1999):

- fornecer suporte para planejamento e controle de projetos de software;
- apoiar a análise de produtividade e qualidade do processo de desenvolvimento de sistemas;
- apoiar a gerência de mudanças de escopo do projeto;
- entender melhor o projeto;
- servir-se como ferramenta de comunicação com os usuários;
- estimar custos de desenvolvimento;
- realizar estudos de *benchmark*;
- facilitar a negociação nos contratos de software; e
- medir artefatos de um sistema.

A métrica FPA tem como princípio básico focar na especificação de requisitos para obter estimativas de tempo, custo, esforço e recursos na fase inicial do projeto ou na manutenção de software, independentemente da tecnologia utilizada para implementação (FUREY, 1997; RIBU, 2001).

A FPA permite uma contagem indicativa no início do desenvolvimento sem conhecer detalhes do modelo de dados. Posteriormente, na fase de projeto, essa contagem passa a ser uma estimativa com maior precisão da complexidade das funções e, ao término do desenvolvimento do software, na implantação, é realizada uma contagem detalhada, obtida a partir do grau de complexidade das funções levantadas no processo funcional, modelo de dados, descrição de telas e relatórios (IFPUG, 2001; LONGSTREET, 2002).

Em geral, as organizações aplicam a FPA como "um método para determinar o tamanho do pacote de software adquirido; para apoiar a análise da produtividade e qualidade; para estimar custos, recursos e esforços de projetos de desenvolvimento e manutenção de software" (GARMUS; HERRON, 2000; HARZAN, 1999).

Além dessas aplicações, Longstreet (2002) destaca outras utilidades da FPA tais como: definir quando e onde fazer reengenharia; estimar casos de testes; entender o as alterações de escopo; ajudar em negociações contratuais; desenvolver um conjunto padrão de métricas; e fazer comparação de software.

O processo de cálculo da FPA é executado em três etapas: (*i*) a determinação dos pontos por função não-ajustados; (*ii*) o cálculo do fator de ajuste; e (*iii*) o ajuste do valor calculado.

A determinação dos pontos por função não-ajustados (*Unadjusted Function Point* – UFP) apresenta a funcionalidade entregue ao usuário, a partir dos requisitos definidos. Envolvem *funções de dados* e *funções transacionais*. As funções de dados são:

- Arquivo Lógico Interno (*Internal Logical File* – ILF) referente a grupos de dados logicamente relacionados ou informações de controle, que são alterados na própria aplicação.
- Arquivo de Interface Externa (*External Interface File* – EIF) referente a grupos de dados logicamente relacionados ou informações de controle, cujo processo de alteração é feito em outra aplicação.

As funções transacionais estão agrupadas da seguinte forma:

- Entrada Externa (*External Input* – EI) que se refere a processos elementares tratando de dados ou informações de controle, que entram pela fronteira da aplicação com o objetivo principal de efetuar manutenção nos ILFs.
- Saída Externa (*External Output* – EO) referente a processos elementares que enviam informações de controle ou dados calculados para o usuário final ou outras aplicações.
- Consulta Externa (*External Inquiry* – EQ) referente a processos elementares que enviam informações de controle ou dados não calculados para o usuário final ou outras aplicações.

O Cálculo do Fator de Ajuste (*Value Adjustement Factor* – VAF) é determinado a partir da soma dos fatores de influência de quatorze características gerais dos sistemas (*General System Characteristics* – GSCs).

A última etapa é o Ajuste do Valor Calculado na primeira etapa, cuja fórmula varia de acordo com o tipo de contagem. No caso de uma aplicação, por exemplo, multiplica-se o UFP pelo VAF.

Os cinco componentes são "pesados" por sua complexidade e somados ao final com pontos de função não ajustados, os UFPs (*Unadjusted Function Point*). Albrecht (1979) não justificou os valores utilizados para os cálculos dos pesos, somente informou que eram "bons resultados" e que eles foram determinados através de debates e tentativas. O total de UFPs deve ser multiplicado pelo Fator de Complexidade Técnica (TCF), que consiste de 14 fatores técnicos e de fatores de qualidade dos requisitos. Segundo Fenton & Pfleeger (1997), o TCF é um direcionador de custos relacionados como tamanho do software.

O método original da FPA foi modificado e refinado diversas vezes. No entanto, o método ainda tem algumas restrições quanto a seu uso. Uma delas é com relação aos pontos de funções que são desenvolvidos para aplicações com processamento de dados, com relação ao uso em aplicações de *real-time* e científico. Outro ponto a ser levado em conta é com relação ao desenvolvimento e pontos de função para sistemas orientados a objetos, onde não tem alcançado o que a técnica propõe (SPARKS, s.d). Os conceitos de orientação a objetos, casos de uso e interface gráfica (GUI) não podem ser convertidos em conceitos de 20 anos atrás como os conceitos abordados pela FPA de elementos de entradas e arquivos lógicos (RULE, 2001).

4.5 MK II FPA

O MK II *Function Point Analysis* (MK II FPA) é uma variação da FPA usada primeiramente no Reino Unido. O método foi proposto por Charles Symons na intenção de melhor contar a complexidade do processamento interno (SYMONS, 1991). Ele observou inúmeros problemas com a contagem original dos pontos de função, dentre eles a excessiva simplificação das escolhas sobre os valores complexos, médios e simples. Isto significava que itens muito complexos não tinham um peso apropriado com suas características.

O MK II *Function Point Analysis*, assim como a FPA, não é uma estimativa adequada para projetos de software orientados a objetos, para projetos de Internet ou para sistemas em *real-time* (SYMONS, 2001),.

Ao invés dos cinco tipos de componentes do método de pontos por função, a MKII enxerga o sistema como uma coleção de "transações lógicas". Cada transação consiste em um componente de entrada, de processamento e de saída. Um tipo de transação lógica é definido como uma única combinação de entrada/processamento/saída disparado por um único evento de interesse do usuário, como, por exemplo: criar cliente; atualizar conta bancária; exibir relatório, etc.

O conceito para a definição de transação lógica é similar ao conceito de caso de uso. Por causa da similaridade entre os conceitos, as duas abordagens (FPA e MKII FPA) podem ser combinadas para produzir melhores estimativas dentro de certas circunstâncias. A MKII modifica a FPA estendendo a lista dos fatores técnicos de ajustes. Esses fatores foram descartados por muito tempo pelo fato de que eles não eram tão significativos em desenvolvimento de software modernos. Tanto os fatores de ajustes da FPA quanto da MKII FPA têm sido descartados como sendo irrealistas (RULE, 2001).

4.6 COSMIC FFP

Nos últimos 15 anos, melhorias têm sido realizadas em busca de um método comum para a medição funcional do tamanho (FSM) para medir os softwares de tempo real. Recentemente, o método COSMIC FFP (*Full Function Points*) foi desenvolvido como uma melhoria dos métodos antecedentes aos pontos de função. Essa nova abordagem foi projetada para trabalhar tanto com aplicações de negócio como com softwares em tempo real (RULE, 2001).

A técnica FFP (*Full Function Points*) é para medir o tamanho de um software baseado em suas funcionalidades. Ela foi proposta para tratar tanto requisitos de softwares embarcados como de tempo real (FFP, 1997).

Utilizando a FFP, os requisitos funcionais são decompostos em processos funcionais e em subprocessos funcionais. Os processos funcionais são equivalentes às transações lógicas da MKII FPA e dos casos de uso. Assim, esta técnica pode ser usada para realizar estimativas de tamanho em software orientado a objetos.

A FFP não leva em consideração para efeito de tamanho os requisitos técnicos e nem os de qualidade, que são medidos separadamente.

4.7 Pontos de Caso de Uso – UCP

A UCP (*Use Case Points*) é uma técnica de estimativa de tamanho de projeto de software orientado a objetos, criada por Karner (1993), com base na FPA, MK II FPA e no e no processo "Objectory", onde foi desenvolvida a técnica de diagramação para o conceito de casos de uso. Posteriormente, Jacobson desenvolveu a "*Object-Oriented Software Engeneering* (OOSE)", metodologia centrada em casos de uso, técnica largamente utilizada na indústria para descrever e capturar os requisitos funcionais de software (RIBU, 2001).

Considerando-se que o modelo de casos de uso foi desenvolvido para capturar os requisitos baseados no uso e na visão dos usuários, faz sentido basear a estimativa de tamanho e recursos de projetos de software nos casos de usos (DAMODARAN; WASHINGTON, 2004).

Na UCP, Karner (1993) explora o modelo e a descrição de casos de uso, substitui algumas características técnicas propostas pela FPA, cria os fatores ambientais e propõe uma estimativa de produtividade.

Como esta técnica é centrada em casos de uso, sua eficácia depende de uma padronização de todos os casos de uso de uma instituição, pois um dos passos para a obtenção das métricas é a contagem de transações por caso de uso. Transações representam um

conjunto de atividades atômicas, e são executadas totalmente ou não, sendo identificadas nos fluxos de eventos dos casos de uso.

A UCP tem contribuído para diminuir algumas dificuldades encontradas pelo mercado em relação à resistência de adoção de métodos de estimativas, porque é uma técnica simples, fácil de usar e rápida de aplicar, quando se têm as informações necessárias para realizar as estimativas (DAMODARAN; WASHINGTON, 2004).

A técnica UCP se baseia em diferentes pesos para a obtenção dos cálculos e é composta por seis etapas a seguir.

4.7.1 Processo para a Contagem de Pontos por Casos de Uso

O processo de contagem da técnica UCP compõe-se de seis etapas: (*i*) classificação dos atores; (*ii*) classificação dos casos de uso; (*iii*) cálculo dos pontos de casos de uso não ajustados; (*iv*) cálculo dos fatores técnicos; (*v*) cálculo dos fatores ambientais; e (*vi*) cálculo dos pontos de caso de uso.

4.7.1.1 Classificação dos Atores

O primeiro passo da UCP é classificar os atores envolvidos em cada caso de uso como conforme a Tabela 4.1. Cada ator é definido como *simples*, *médio*, ou *complexo*.

Tabela 4.1 – Classificação dos atores

Complexidade do Ator	Descrição	Peso
Simples	Aplicação com APIs definidas.	**1**
Médio	Aplicação com interface baseada em protocolo ou interação de usuário baseado em linhas de comandos.	**2**
Complexo	Interação de usuário através de Interface Gráfica ou página Web.	**3**

O peso total dos atores do sistema, UAW (*Unadjusted Actor Weight*), é calculado pela soma dos produtos dos atores de cada tipo pelo respectivo peso.

Um sistema com três tipos de usuários, por exemplo, um gerente e um usuário (atores complexos), acessado por outro sistema através de um protocolo de comunicação como TCP/IP (ator médio) teria o valor de UAW = 8, conforme Tabela 4.2.

Tabela 4.2 – Exemplo de pesos de atores

Atores	Quantidade	Peso	Peso Total
Médio	1	2	1 * 2 = 2
Complexo	2	3	2 * 3 = 6
TOTAL (UAW = ∑ Peso Total dos Atores)			**8**

4.7.1.2 Classificação dos Casos de Uso

Depois de calculado o peso dos atores, deve-se calcular o peso bruto dos casos de uso, UUCW (*Unadjusted Use Case Weight*). Cada caso de uso é definido como *simples*, *médio*, ou *complexo*, dependendo do número de transações contidas na descrição do caso de uso (RIBU, 2001).

Uma outra maneira de se calcular o peso dos casos de uso do sistema é levar em consideração o número de classes de análise envolvidas no processo. O cálculo, neste caso, é realizado da mesma forma que na abordagem anterior, e pode ser aplicado quando já for possível determinar que classes implementam um caso de uso específico (SCHNEIDER & WINTERS, 2001).

Um caso de uso simples é implementado com até quatro classes, um caso de uso médio tem de cinco a dez classes e um caso de uso complexo pode ter mais de 10 classes associadas àquele caso de uso. Classes de análise representam um modelo conceitual para "coisas" em um sistema, que possuem responsabilidades e comportamento. Estas classes podem ser utilizadas para representar os objetos que o sistema deve dar sustentação, sem decidir o quanto elas sustentarão questões de hardware e software (RATIONAL CORPORATION, 2003).

Por transação entende-se um conjunto de atividades atômicas, as quais são executadas completamente ou não (RIBU, 2001), ou seja, uma série de processos que devem, garantidamente, ser realizados em conjunto ou cancelados em sua totalidade. O total de transações do caso de uso corresponde, no fluxo de eventos, à quantidade de transações do fluxo básico e dos fluxos alternativos.

O número de transações (ou o número de classes) deve ser multiplicado pelo respectivo peso de cada caso de uso. A Tabela 4.3 mostra a classificação de casos de uso.

Tabela 4.3 – Classificação dos casos de uso

Complexidade Caso de Uso	Descrição	Peso
Simples	Tem até 3 transações incluindo os passos alternativos ou deve ser realizado com menos de 5 classes de análise.	**5**
Médio	Tem de 4 a 7 transações incluindo os passos alternativos ou deve ser realizado com 5 a 10 classes de análise.	**10**
Complexo	Tem acima de 7 transações incluindo os passos alternativos ou deve ser realizado com pelo menos 10 classes de análise.	**15**

A complexidade dos casos de uso é obtida pela soma de cada tipo de caso de uso multiplicado pelo seu peso, conforme Tabela 4.4.

Tabela 4.4 – Exemplo de pesos de casos de uso

Complexidade	Quantidade	Peso	Peso Total
Simples	03	05	15
Médio	18	10	180
Complexo	02	15	30
TOTAL (UUCW = ∑Peso Total dos Casos de Uso)			**225**

4.7.1.3 Pontos de Casos de Uso Não Ajustados

Para se calcular os pontos de caso de uso não ajustados (UUCP- *Unadjusted Use Case Points*) no projeto, basta somar o cálculo da complexidade de atores com o cálculo da complexidade dos casos de uso, conforme a *Eq*. 4.3.

$$\mathbf{UUCP = \sum UAW + \sum UUCW} \qquad (Eq.\ 4.3)$$

4.7.1.4 Cálculo dos Fatores Técnicos

Os ajustes são calculados de forma similar aos adotados pela técnica Análise por Pontos de Função, para que sejam levados também em consideração nos cálculos das estimativas dos fatores técnicos e fatores ambientais.

Será mostrado primeiramente o cálculo de ajuste para fatores técnicos, que cobre uma série de requisitos funcionais do sistema. Os fatores de complexidade técnica, TCF (*Technical Complexity Factor*), são exibidos na Tabela 4.5 juntamente com seus respectivos pesos. Eles podem receber notas que variam de 0 a 5. O valor 0 (zero) indica que este quesito é irrelevante para o projeto, de pouca criticidade e baixa complexidade; o valor 3 indica influência moderada; e o valor 5 indica uma forte influência, alta criticidade e complexidade.

Tabela 4.5 – Fatores de complexidade técnica (RIBU, 2001)

Fator	Descrição	Peso
T1	Sistemas Distribuídos.	2.0
T2	Desempenho da Aplicação.	1.0
T3	Eficiência do Usuário-Final (*on-line*).	1.0
T4	Processamento Interno Complexo.	1.0
T5	Reusabilidade do código em outras aplicações.	1.0
T6	Facilidade de Instalação.	0.5
T7	Usabilidade (facilidade operacional).	0.5
T8	Portabilidade.	2.0
T9	Facilidade de Manutenção.	1.0
T10	Concorrência.	1.0
T11	Características especiais de segurança.	1.0
T12	Acesso direto para terceiros.	1.0
T13	Facilidades especiais de treinamento.	1.0

O TCF é calculado multiplicando-se o valor de cada fator (T1 – T13) pelo respectivo peso, e depois se calcula o somatório de todos esses valores no chamado TFator. O TCF é dado pela *Eq*. 4.4.

$$\mathbf{TCF = 0.6 + (0.01 * TFator)} \qquad (Eq.\ 4.4)$$

Será apresentado um exemplo da análise dos fatores técnicos de um sistema Web fictício de agenda de compromissos e telefones, na Tabela 4.6.

Tabela 4.6 – Exemplo da análise dos fatores técnicos em um sistema *web*

Fator	Descrição	Peso	Nota	Peso Total
T1	O sistema é distribuído	2,0	5	10
T2	O sistema deve ser otimizado	1,0	5	5
T3	A eficiência do Usuário-Final não é extremamente necessária nesse tipo de aplicação	1,0	3	3
T4	A aplicação não possui processamento interno complexo	1,0	3	3
T5	É uma boa prática	1,0	3	3
T6	Precisa ser fácil para o usuário final	0,5	5	2,5
T7	Precisa ser fácil para o usuário final	0,5	5	2,5
T8	Não há necessidade	2,0	0	0
T9	É uma boa prática	1,0	3	3
T10	Por ser ambiente Web, há necessidade de controle de concorrência eficiente	1,0	5	5
T11	Necessidade de autenticação de usuário	1,0	5	5
T12	Não haverá acesso direto para terceiros	1,0	0	0
T13	Não há necessidade de treinamento, por ser bastante simples e intuitivo	1,0	0	0
TFator				**42**

O cálculo dos pontos relativos aos fatores técnicos ficaria da seguinte forma:

TFator = ∑Factor

= (5*2) + (5*1) + (3*1) + (3*1) + (3*1) + (5*0,5) +(5*0,5 +

(0*2) + (3*1) + (5*1) + (5*1) + (0*1) + (0*1)

= 42

TCF = 0,6 + (0,01 * TFactor)

= 0,6 + (0,01 * 42)

= **1,02**

4.7.1.5 Cálculo dos Fatores Ambientais

O cálculo dos fatores ambientais (*Environmental Factor* – EF) diz respeito aos requisitos não-funcionais associados ao projeto, tais como a experiência da equipe, a estabilidade do projeto e a motivação dos programadores.

Da mesma forma como o fator técnico, o fator ambiental utiliza-se do mesmo mecanismo de peso, onde cada peso deve ser multiplicado pelo valor de 0 a 5. Para fatores F1 a F4, 0 (zero) significa nenhuma experiência no assunto, 5 significa excelência no assunto, e 3 um conhecimento mediano sobre o assunto.

Para o fator F6, 0 significa requisitos instáveis, 5 imutáveis, e 3 poucas mudanças. Para o fator F7, 0 significa nenhum colaborador por meio período, 5 significa que todos os

colaboradores trabalham meio período, e 3 que alguns colaboradores trabalham meio período. Para o fator F8, 0 significa uma linguagem com pouca ou nenhuma dificuldade de programação, 3 significa com uma complexidade média, e 5 com uma alta complexidade e dificuldade. A Tabela 4.7 mostra os fatores ambientais previstos e seus respectivos pesos.

Tabela 4.7 – Fatores ambientais (RIBU, 2001)

Fator	Descrição	Peso
F1	Familiaridade com o RUP ou outro processo formal	**1,5**
F2	Experiência com a aplicação em desenvolvimento	**0,5**
F3	Experiência com orientação a objeto	**1**
F4	Capacidade de análise	**0,5**
F5	Motivação	**1**
F6	Requerimentos estáveis	**2**
F7	Colaboradores de meio período	**-1**
F8	Dificuldade na linguagem de programação	**-1**

No caso do valor do Fator Ambiental (EF), determinar que um fator tem influência alta indica que este fator está presente no projeto como um todo; atribuir um valor 0 significa que o fator não está presente no processo de desenvolvimento.

O Fator Ambiental (EF) é calculado pela *Eq*. 4.5.

$$\mathbf{EF = 1.4 + (- 0.03 \times EFator)} \qquad (Eq.\ 4.5)$$

Usando ainda como exemplo o mesmo sistema Web de agenda de compromissos e telefones, a Tabela 4.8 mostra o cálculo desses fatores ambientais.

Tabela 4.8 – Exemplo da análise dos fatores ambientais em um sistema *web*

Fator	Descrição	Peso	Nota	Peso Total
F1	Familiaridade com o RUP	1,5	2	3
F2	Experiência em desenvolvimento	0,5	5	2,5
F3	Experiência em OO	1	4	4
F4	Capacidade de análise	0,5	4	2
F5	Motivação	1	5	5
F6	Pode haver mudanças no negócio	2	1	2
F7	Colaboradores em meio período	-1	0	0
F8	Dificuldade na linguagem de programação	-1	3	-3
TFator				**15,5**

O cálculo dos pontos relativos aos fatores ambientais ficaria da seguinte forma:

EFator = (2*1,5) + (5*0,5) + (4*1) + (4*0,5) + (5*1) + (1*2) + (0*-1) + (3*-1)

= 15,5

EF = 1,4 + (-0,03 * EFactor)

= 1,4 + (-0,03 * 15,5)

= **0,935**

4.7.1.6 Cálculo dos Pontos de Caso de Uso

Depois de calculado os valores UUCP (*Unadjusted Use Case Points*), TCF (*Technical Complexity Factor*) e EF (*Environmental Factor*), pode-se, finalmente, calcular o UCP (*Use Case Points*) ajustado, que consiste no tamanho total do sistema. Para isso utiliza-se a *Eq*. 4.6.

$$\mathbf{UCP = UUCP * TCF * EF} \qquad (Eq.\ 4.6)$$

4.7.2 Estimativa de Esforço do Projeto

Para a obtenção da estimativa de esforço total do projeto, a UCP utiliza um fator de produtividade (PROD). O fator de produtividade pode ser calibrado de acordo com a produtividade da equipe de um projeto e, em geral, é dado em horas.

Karner (1993) propôs um fator médio de 20 horas de trabalho por ponto de caso de Uso (UPC). Porém, experiências demonstram uma variação entre 15 e 30 horas por ponto.

Schneider e Winters (2001) sugerem um refinamento na técnica de Karner (1993), baseado no nível de experiência da equipe e na estabilidade do projeto. Eles sugerem que certos fatores têm uma influência mais direta na média de horas. Esta técnica consiste em:

- contar o número de fatores entre F1 e F6 que receberam valores acima de 3;
- contar o número de fatores em F7 e F8 que receberam valores menores que 3.
- somar as duas contagens e o total é uma sugestão de horas por ponto de caso de uso a ser adotada no projeto, segundo a média:
  - se o total é menor que 2, propõe-se 20 horas por UCP;
  - se o total é entre 3 e 4, o valor será 28 horas por UCP;
  - valores maiores que 4, o valor será 36 horas por UCP. Nesse caso, recomenda-se que sejam feitas modificações no projeto. Esses fatores medem o nível da equipe e a estabilidade do projeto. Valores acima de 4 sugerem instabilidade no projeto ou necessidade de treinamento da equipe.

Segundo Ribu (2001), até mesmo pequenos ajustes nos fatores, de meio ponto, por exemplo, podem fazer uma grande diferença. Em um exemplo de estudo de caso, ajustando o fator ambiental de experiência em OO de 3 para 2,5, houve um aumento de 4.580 horas. Isto demonstra que, se os valores não forem informados de maneira precisa, os resultados não serão precisos. Na UCP, o cálculo do esforço total do projeto é dado na *Eq*. 4.7.

$$\textbf{Esforço} = \textbf{UCP} * \textbf{PROD} \qquad (Eq.\ 4.7)$$

4.8 Trabalhos Relacionados ao Uso da Técnica UCP

A UCP, apesar de sua simplicidade, ainda é uma técnica que não tem a utilização da FPA. Algumas razões para o baixo uso dessa técnica foram indicadas por DAMODARAN & WASHINGTON (2004) tais como:

- a UCP é uma técnica relativamente nova;
- ainda não foi incorporada em ferramentas populares de estimativa de software;
- o modelo de caso de uso, apesar de ser um método muito utilizado para a descrição de requisitos, ainda não possui bons históricos de produtividade.

Experiências de uso do UCP na indústria foram realizadas por Anda *et al*. (2001), por Ribu (2001) e por algumas empresas como a Rational, SUN e IBM (PROBASCO, 2002).

No Brasil, algumas empresas privadas de desenvolvimento de software têm adotado a UCP para estimativa de tamanho de projetos de software ou para estudos experimentais. Algumas dessas experiências já foram publicadas por Valença (2003), Cunha & Almeida (2003) e Feitosa & Jansen (2003).

Diversos autores já experimentaram esta métrica com e sem o uso dos fatores técnicos de ajuste, computaram a estimativa de esforço, propuseram um formulário padrão para descrição de casos de uso e definiram regras para atribuir valores aos fatores ambientais (LONGSTREET, 2000; SCHNEIDER *et al*., 2001; ANDA *et al*., 2001; ANDA, 2002; RIBU, 2001).

O método de Karner (1993) foi aplicado por Anda *et al*. (2001) em três projetos de uma companhia de desenvolvimento de software da Noruega, Suécia e Finlândia para comparar as estimativas realizadas através do UCP com as estimativas reais dos projetos elaboradas por gerentes especialistas. Os resultados mostraram que a estimativa realizada através de UCP foi próxima à estimativa real realizada pelos desenvolvedores experientes. Estes autores observaram que alguns aspectos da estrutura dos modelos de casos de uso tiveram impacto nas estimativas tais como: (*i*) o uso de generalização entre os atores; (*ii*) a

contagem dos casos de uso estendidos e incluídos; (*iii*) o nível de detalhe da descrição dos casos de uso; (*iv*) as dificuldades para atribuição de valores aos fatores técnicos e ambientais e; (*v*) a escolha da taxa de produtividade por UCP. Um formulário padrão para facilitar a descrição de casos de uso foi definido por ANDA *et al.* (2001).

Estudos realizados por ANDA *et al*. (2001) revelaram que o modelo de casos de uso tem um impacto forte na estimativa; pode ser utilizado com sucesso para estimar esforço em desenvolvimento de software; e que o método de Karner (1993) pode apoiar os especialistas no processo de estimativa baseado na FPA.

Apesar de Karner (1993) não recomendar a contagem dos casos de uso estendidos e incluídos, ANDA *et al.* (2001) acham que os mesmos devem ser incluídos na contagem, para evitar estimativa abaixo da realidade, principalmente se as funções descritas nesses casos de uso são essenciais e implementadas.

Anda (2002), em outro trabalho, investigou a UCP mediante a realização de cursos de modelagem de caso de uso para 37 profissionais experientes em desenvolvimento de software. Fez uma comparação entre a medição dos especialistas em desenvolvimento e em estimativa de software, com a medição realizada por técnicos inexperientes com o domínio da aplicação e da tecnologia adotada para desenvolvimento. Neste estudo, foi utilizado o formulário padrão definido no seu trabalho anterior para descrever os casos de uso.

Ribu, (2001) também utilizou o método de Karner (1993) para estimar o tamanho de projetos de estudantes e da indústria. Ele testou a UCP com e sem o uso dos fatores técnicos de ajuste, computou a estimativa de esforço, propôs um formulário padrão para descrição de casos de uso, e definiu regras para atribuir valores aos fatores ambientais. Além disso, propôs a análise da complexidade do caso de uso através do diagrama de seqüência e das classes que o implementam, quando não houver sua descrição.

Ribu, (2001) ressaltou que a alternativa acima pode levar à contagem de classes de projeto e comprometer o resultado da estimativa. As conclusões desse estudo foram as seguintes: (*i*) a técnica UCP teve baixa variação entre o valor estimado e o real; (*ii*) sugeriu descartar os fatores de ajustes técnicos e manter os fatores ambientais; (*iii*) contar os casos de uso incluídos e estendidos; e (*iv*) utilizar um formulário padrão para descrição de casos de uso.

4.9 Conclusão

Este capítulo apresentou os principais conceitos relacionados às métricas de estimativas e ao processo de estimativa de software, cujo propósito era estabelecer uma forma

para estimar o tamanho, o esforço, o prazo e o custo de um projeto de software. Foram apresentados também os principais métodos e técnicas para estimar um projeto de software, tendo sido detalhada a técnica UCP, foco deste trabalho.

No próximo capítulo, será apresentada uma proposta de extensão da UCP, a chamada TUCP, com objetivo principal de refinar a precisão da UCP.

Capítulo **5**

TUCP – uma Extensão da UCP

Este capítulo propõe uma extensão da técnica de Pontos de Caso de Uso (UCP), objetivando um refinamento das estimativas dessa técnica, e a utilização das mesmas ao longo das etapas do processo de desenvolvimento de software.

É cada vez maior o número de organizações que se preocupam com a melhoria da qualidade de seus produtos de software, especialmente em decorrência dos custos efetivos e do cumprimento de cronogramas desses produtos, o que depende fortemente do gerenciamento eficiente e efetivo de um processo de desenvolvimento de software que seja eficaz.

Entretanto, para que isto se concretize é essencial a adoção de normas, técnicas, ferramentas e modelos apregoados pela engenharia de software como: NBR ISO/IEC 12.207 (ABNT, 1998), ISO/IEC 15.504 (ISO, 2003); CMMI (SEI, 2002) e PMBOK (2004).

Um gerenciamento eficiente de projetos de software deve partir de um bom plano de projeto, que possa atender às exigências do cliente e esteja baseado em estimativas precisas de tamanho, esforço, prazos e custos.

Além de subsidiar o planejamento do projeto, a elaboração de estimativas facilita o relacionamento entre cliente e fornecedor, permitindo o gerenciamento de riscos, o controle de cronogramas, e possibilita o conhecimento da produtividade da equipe, o que também beneficia o gerenciamento e a qualidade de possíveis contratos de terceirização de projetos de software (HAZAN, 1999; GARMUS; HERRON, 2000 E LONGSTREET, 2002).

A precisão de estimativas de tamanho depende de informações que nem sempre estão disponíveis na concepção de um projeto. No entanto, essas estimativas são necessárias desde o início do projeto, para a elaboração de contratos ou a determinação da viabilidade desse projeto em termos da análise de custos e benefícios. Portanto, é necessária a obtenção de estimativas mais precisas desde o princípio do desenvolvimento do projeto, bem como o gerenciamento das mesmas ao longo de seu ciclo de vida.

É reconhecido na comunidade científica que as estimativas de software não são fáceis de ser realizadas ao longo de seu processo de desenvolvimento, principalmente se no início do projeto. Neste sentido, a UCP tem contribuído para a diminuição de algumas dificuldades encontradas em outras estimativas existentes. Isto porque é uma técnica simples, fácil e rápida de ser usada, que se amolda à orientação a objetos (OO), utilizando a abordagem de casos de uso para expor as funcionalidades do sistema.

No entanto, a UCP ainda não é uma técnica amplamente utilizada. Algumas organizações que a empregaram, por exemplo, em projetos com muitos casos de usos classificados como complexos, dizem ter tido resultados de estimativas com margem de erro superior ao desejável. Além disso, essa técnica não fornece uma visão detalhada das estimativas por etapa do ciclo de vida. Neste contexto, surgiu a proposta de extensão da UCP (*Use Case Points*) para a TUCP (*Technical Use Case Point*s) no presente trabalho.

5.1 Objetivos da TUCP

A idéia de se estender a UCP para a TUCP é conseguir uma maior precisão na contagem de pontos de caso de uso e fornecer uma visão mais detalhada das estimativas (tamanho, esforço, prazo e custos) por etapas do ciclo de vida do projeto.

A TUCP aborda os seguintes pontos:

- Propor um *Guia para a elaboração de casos de uso*, influenciando fortemente no cálculo do tamanho dos casos de uso.
- Detalhar o conceito de transação no contexto de casos de uso, por afetar diretamente o cálculo dos pontos de casos de uso não ajustados (capítulo 4).
- Refinar a contagem de pontos por casos de uso complexos, para evitar possíveis subestimativas, especialmente quando o número de transações é muito grande.
- Desatrelar os fatores técnicos de ambiente (EF) do cálculo do tamanho, por se entender que o tamanho é uma grandeza física que não deveria ter seu valor alterado em função dos fatores técnicos de ambiente.
- Considerar apenas no cálculo do esforço os EFs, juntamente com o fator de produtividade (de forma semelhante à UCP).
- Detalhar o cálculo do esforço nas etapas do ciclo de vida.
- Granularizar o cálculo do tamanho do projeto por cada etapa do processo de desenvolvimento e por caso de uso. O somatório de todos esses tamanhos será o tamanho final para desenvolver o projeto, não incluindo os esforços indiretos.

Nas próximas seções, cada ponto levantado será descrito detalhadamente.

5.2 Guia de Especificação de Casos de Uso

De forma semelhante a UCP, a TUCP também utiliza os conceitos da UML (*Unified Modeling Language*) para modelar os requisitos funcionais de sistemas orientados a objetos (OO), através de casos de uso. Mesmo os sistemas que não sejam OO, podem ter seus requisitos modelados utilizando a abordagem de casos de uso.

Várias têm sido as propostas de como se construir casos de uso efetivos, através de formulários e regras para sua descrição (ANDA *et al*., 2001; RIBU, 2001; ANDA, 2002; OMG, 2003). No entanto, ainda não há uma padronização largamente aceita para a descrição de casos de uso (FOWLER, 2005), especialmente por esta ser de natureza subjetiva e depender do grau de detalhamento e do conhecimento das informações disponibilizadas.

Neste trabalho, é proposto um *Modelo de Especificação de Casos de Uso* (Apêndice A), a partir de pesquisas na literatura sobre o tema, pois a forma de especificar casos de uso é um fator preponderante no cálculo da estimativa de tamanho do software, usando-se a técnica UCP.

A estimativa de tamanho de software em uma organização, que utilize a abordagem UCP, somente será suficientemente precisa e passível de ser calculada, se houver nessa organização um modelo que padronize a especificação de casos de uso (Apêndice A), que seja seguido adequadamente pelos projetos de software. Além disso, faz-se necessária a elaboração de um documento que oriente como preencher esse modelo de especificação, o que permite se chegar a um nível de padronização ainda maior dentro da organização. Neste contexto, foi proposto um *Guia para Elaborar a Especificação de Caso de Uso* (Apêndice B).

Uma especificação de casos de uso deverá ser elaborada para cada caso de uso do sistema. Assim, ter-se-á o mesmo número de especificações de casos de uso quantos forem os casos de uso do sistema, facilitando, posteriormente, na contagem das transações de cada caso de uso, que é uma das etapas chaves para o cálculo das estimativas de tamanho.

5.3 Contagem das Transações de Casos de Uso

A quantidade de transações em um caso de uso determina sua complexidade, podendo classificá-lo como *simples*, *intermediário* e *complexo*. A identificação das transações de um caso de uso não é trivial devido ao grau de subjetividade existente neste tipo de contagem. É muito importante o entendimento claro do conceito de uma transação, no contexto de caso de uso.

Segundo Anda (2002), as transações representam um conjunto de atividades atômicas, e são executadas totalmente ou não, sendo identificadas nos fluxos de casos de uso. Uma transação é um evento que ocorre entre um ator e o sistema.

No contexto deste trabalho, definimos transação como sendo cada passo dos fluxos de eventos (básico e ou alternativo) de casos de uso, que deva ser executado por completo, ou a realização de algum processamento complexo nesses fluxos de eventos.

Por transação, entende-se uma série de processos que devem, garantidamente, ser realizados em conjunto ou cancelados em sua totalidade, caso não seja possível concluir o processamento (FREIRE, s.d).

De acordo com Schneider e Winters (2001), transação é definida como um conjunto de atividades atômicas, as quais são executadas completamente ou não. Quando houver a mesma transação em vários diagramas de seqüência como, e.g. *logging* ou procedimentos de segurança, a transação deve ser contada apenas uma vez, porque a funcionalidade é implementada apenas uma vez e reutilizada em outros casos de uso (RIBU, 2001).

Em muitos trabalhos que descrevem a técnica UCP, vários deles apresentam somente os conceitos sobre transações de casos de uso. Somente a conceituação não é suficiente para que a classificação de um caso de uso seja realizada de maneira correta. Ribu (2001) sugeriu as seguintes regras para identificar casos de uso e suas respectivas transações:

- contar os casos de uso estendidos e incluídos separadamente apenas uma vez e não como uma transação de cada caso de uso que o utiliza;
- identificar as classes de análise (não as de projeto) que implementam as funções do caso de uso, para auxiliar na determinação da complexidade do caso de uso. Essas classes podem ser identificadas na descrição dos casos de uso, nos diagramas de seqüência ou no diagrama de classes de análise;
- comparar as transações (descritas no caso de uso) com as transações do diagrama de seqüência e com o número de classes de análise utilizadas para implementar o caso de uso, para verificar se a complexidade do caso de uso foi definida corretamente;
- quando não houver a especificação do caso de uso, deve-se utilizar o diagrama de seqüência para contar as transações (apenas aquelas que indicam o que fazer e não as que indicam o como fazer) ou as classes que implementam o caso de uso.

Neste contexto, o presente trabalho propõe alguns direcionamentos a seguir para auxiliar na identificação de transações em casos de uso e possibilitar que sua medição seja adequada à técnica TUCP.

O que deve ser contado com uma transação:

- passos do fluxo de eventos do caso de uso que contenham campos de entrada possuindo valores passíveis de escolha originados de leitura de dados (listas de opções, *combos* e *grids*), por exemplo: "A aplicação exibe os nomes dos assinantes de uma companhia telefônica";
- passos do fluxo de eventos do caso de uso que apresentem retorno de consultas com filtros preenchidos por buscas em bancos de dados, por exemplo: "A aplicação exibe uma lista contendo os nomes da rua de acordo com o CEP selecionado";
- passos do fluxo de eventos do caso de uso que proporcionem validações complexas de negócio (relacionamentos com outras entidades, verificações de existência, etc);
- passos do fluxo de eventos do caso de uso que contenham uma geração de relatório são considerados como uma transação, e cada filtro das consultas apresentados no relatório será considerado uma outra transação; assim, a quantidade de filtros será a quantidade de transações;
- passos onde é necessária a transformação de extensões entre arquivos. Por exemplo: passar um arquivo HTML passar para XML;
- passos do fluxo de eventos do caso de uso que apresentem funcionalidades de consultas auxiliares como casos de uso a parte (extensão), por exemplo, telas *pop-up*;
- fluxos alternativos do caso de uso tiver uma quantidade de validações maior ou igual a 5, conta-se como transação e se a quantidade de validações for maior que 10, conta-se uma transação a cada grupo de 5 validações; e
- passos do fluxo de eventos do caso de uso onde existirem validações simples (obrigatoriedade, formato, etc) de campo de entrada de dados são considerados como uma única transação se a quantidade de validações for menor ou igual a 10; se a quantidade de validações for maior que 10, conta-se uma transação a cada grupo de 5 validações. Isto foi obtido através de experimentos realizados nos ativos da organização onde se realizou o estudo de caso.

O que não deve ser contado como transação:

- passos do fluxo de eventos que descrevam o início e o fim do caso de uso, por exemplo: "O caso de uso se inicia quando o usuário abre uma tela de filtro" ou "O caso de uso se encerra";

- passos do fluxo de eventos que detalhem a interação entre o sistema e o ator, por exemplo, "O usuário pressiona confirmar" ou "o sistema solicita ao usuário informar a operação (incluir, alterar, excluir)";
- passos do fluxo de eventos que solicitem escolhas com valores fixos (sem leitura de dados);
- passos do fluxo de eventos que façam leituras auxiliares de dados que já tenham sido realizadas em outros fluxos do mesmo caso de uso; e
- fluxos alternativos do caso de uso que são apenas validações simples e que contenham mensagens de erro. Por exemplo: "Se após o passo 5 do fluxo básico, ocorrer um erro de acesso à base de dados, a aplicação exibe a mensagem 'Erro no acesso à base de dados'".

A contagem das transações está exemplificada no Anexo A. Cada passo do caso de uso considerado como uma transação foi indicado por meio da representação "**(1T)**", no final de sua descrição.

O grau de precisão da TUCP é dependente de se ter na organização um modelo adequado para a elaboração de especificação de caso de uso (seção 5.2), como também é dependente do entendimento claro do que seja o conceito de transação. Estas questões influenciam diretamente no cálculo do tamanho do sistema.

5.4 Estimativa de Tamanho da TUCP

As estimativas de tamanho são o primeiro passo a ser seguido, pois as outras estimativas (esforço, prazo e custo) serão dependentes da estimativa de tamanho. A técnica de contagem da TUCP é uma extensão da técnica de Karner (1993).

No cálculo do tamanho do software na UCP, os fatores de ambiente (EF) são considerados. Assim sendo, o uso desses fatores pode levar a diferentes tamanhos, dependendo da equipe ou empresa que desenvolver um dado projeto. Assim, entende-se que apenas os fatores técnicos e não-técnicos deveriam ser tidos na estimativa de tamanho, evitando gerar dependência do tamanho com características da equipe de desenvolvimento e da empresa. Todavia, os fatores ambientais têm sua influência efetiva no cálculo das estimativas de esforço. Neste contexto, os fatores ambientais (EF) não serão levados em consideração para efeito de cálculo na estimativa de tamanho da TUCP, mas serão considerados no cálculo das estimativas de esforço.

O processo de contagem de TUCP compõe-se das cinco etapas a seguir:

i. Contagem dos atores (UAW);

ii. Contagem dos casos de uso (TUUCW);

iii. Cálculo dos pontos de casos de uso não ajustados (TUUCP);

iv. Cálculo dos fatores de complexidade técnica (TCF);

v. Cálculo dos pontos de caso de uso técnicos (TUCP).

5.4.1 Contagem dos Atores

A maneira para classificar os atores envolvidos em cada caso de uso é semelhante ao utilizado na técnica UCP (KARNER, 1993).

O peso total dos atores do sistema é calculado pela soma dos produtos dos atores de cada tipo pelo respectivo peso, UAW (*Unadjusted Actor Weight*), de acordo com os valores da Tabela 5.1.

Tabela 5.1 - Classificação dos atores

Complexidade do Ator	Descrição	Peso (UAW)
Simples	Aplicação com APIs definidas	**1**
Médio	Aplicação com interface baseada em protocolo ou interação de usuário baseado em linhas de comandos	**2**
Complexo	Interação de usuário através de interface gráfica ou página Web.	**3**

5.4.2 Contagem dos Casos de Uso

A contagem dos casos de uso foi alterada em relação à técnica original (KARNER, 1993), para refletir uma realidade já observada em várias organizações, quando se tratava da contagem de casos de uso complexos.

Através de experiências vivenciadas em dezenas de projetos de software utilizando-se a técnica UCP, em uma organização certificada SW-CMM, nível 2, onde foram realizados os estudos de caso deste trabalho, constataram-se atrasos na entrega do produto final para o cliente. Esses atrasos, em geral, estavam relacionados a projetos que continham casos de uso complexos, especialmente aqueles com um grande número de transações (maior que 12 transações).

Com base nesta questão, uma adaptação no cálculo dos pesos para os casos de uso complexos foi elaborada na Tabela 5.2, gerando-se assim o TUUCW (*Technical Unadjusted Use Case Points Weight*).

Tabela 5.2. Contagem dos casos de uso

Tipo de Caso de Uso	Descrição	Peso (TUUCW)
simples	até 3 transações incluindo os passos alternativos	5
médio	de 4 a 7 transações incluindo os passos alternativos	10
complexo	de 8 a *t* transações incluindo os passos alternativos	15
n-complexo	acima de *t* transações	P_X

Os casos de uso com até *t* transações serão calculados da mesma maneira apresentada por Karner (1993). Acima de *t* transações, o tipo de caso de uso será denominado de *n*-complexo. O cálculo do peso dos casos de usos *n*-complexos é exibido nas equações: *Eq*. 5.1,e *Eq*. 5.2.

$$\text{TUUCW} = 15n + p \qquad (Eq.\ 5.1)$$

$$n = T / t \qquad (Eq.\ 5.2)$$

Onde: T = número de transações do caso de uso; p = é o peso obtido, quando o resto (r) da divisão de T / t é aplicado ao peso original (*simples*, *médio*, e *complexo*), (Tabela 5.2).

Neste contexto, se $r = 0$, $p = 0$; se $r \in [1, 3]$, $p = 5$; se $r \in [4, 7]$, $p = 10$; e se $r \in [8, (t - 1)]$, $p = 15$.

Por exemplo: um caso de uso com 13 transações (T = 13), e com $t = 11$, tem então, $n = 13 / 11$, e $r = 2$. Como $r \in [1, 3]$, então $p = 5$. Assim:

$$\text{TUUCW} = 15 * 1 + 5 = 20$$

A definição do valor de *t* poderá depender das características da organização, ou até mesmo das características de um dado tipo de projeto. Recomenda-se a calibração do valor de *t*, após ter sido levantado um histórico de (tipos) projetos concluídos, que possuam casos de uso do tipo *n*-complexo. Essa calibração deverá ser realizada baseada no esforço real de implementação do conjunto de projetos considerados.

Após a definição do valor de *t*, o peso de um caso de uso *n*-complexo vai sendo incrementado em um ciclo do tipo (3; 4; *k*). Isto significa dizer que a cada (ft + 3), ((ft + 3) + 4), (((ft + 3) + 4) + k), ... transações, e assim por diante, o valor do peso é modificado. O valor de $f \in \mathrm{N}$ e que *k* é dado pela *Eq*. 5.3.

$$k = t - 7 \qquad (Eq.\ 5.3)$$

Na organização SW-CMM, nível 2, onde foram realizados os experimentos durante o período de aproximadamente dois anos, envolvendo uma média de sessenta projetos, o valor de *k* que se mostrou mais apropriado foi $k = 4$. Experimentos variando o valor de *k* entre 4 a 7

foram empreendidos nesses projetos. A partir da análise dos dados entre estimativas planejadas e realizadas foi verificado (depois de várias simulações), que o valor mais apropriado era $k = 4$. Na prática, é como se tivéssemos um caso de uso *n-complexo* transformado em *n* casos de uso *complexos* (com até 11 transações cada um deles).

No entanto, percebeu-se que alguns dos projetos que participaram dos experimentos tinham características peculiares (e.g.: utilização de *frameworks* de desenvolvimento, forte grau de reuso). Para esses projetos, um valor de $k > 4$ indicou resultados mais próximos do esforço real despendido nesses projetos.

5.4.3 Cálculo dos Pontos de Casos de Uso não Ajustados

Para se calcular os pontos de caso de uso não ajustados – TUUCP (*Technical Unadjusted Use Case Points*) – basta efetuar o somatório entre a complexidade de atores (Tabela 5.1) e o cálculo da complexidade dos casos de uso (Tabela 5.2), conforme a *Eq*. 5.4.

$$\text{TUUCP} = \sum \text{UAW} + \sum \text{TUUCW} \qquad (Eq.\ 5.4)$$

5.4.4 Cálculo dos Fatores de Complexidade Técnica

Os fatores de complexidade técnica (TCF) são calculados de forma similar aos apresentados pela técnica UCP (seção 4.7.1.4).

Da mesma maneira que a técnica original da UCP, os fatores técnicos cobrem uma série de requisitos funcionais do sistema. Os TCFs são exibidos na Tabela 5.3 com seus respectivos pesos. Podem receber notas que variam de 0 a 5, o valor 0 (zero) indicando que este item é irrelevante para o projeto, de pouca criticidade e baixa complexidade; o valor 3 indicando influência moderada; e o valor 5 uma forte influência, alta criticidade e complexidade.

Tabela 5.3 - Fatores de complexidade técnica (Ribu, 2001)

Fator	Descrição	Peso
T1	Sistemas Distribuídos	2.0
T2	Desempenho da Aplicação	1.0
T3	Eficiência do Usuário-Final (*on-line*)	1.0
T4	Processamento Interno Complexo.	1.0
T5	Reusabilidade do código em outras aplicações	1.0
T6	Facilidade de Instalação	0.5
T7	Usabilidade (facilidade operacional)	0.5
T8	Portabilidade	2.0
T9	Facilidade de Manutenção	1.0
T10	Concorrência	1.0
T11	Características especiais de segurança	1.0
T12	Acesso direto para terceiros	1.0
T13	Facilidades especiais de treinamento	1.0

O TCF é obtido através da *Eq*. 5.5, onde TFator corresponde ao somatório dos produtos entre o peso e a nota atribuída de cada fator.

$$TCF = 0{,}6 + (0{,}01 * TFactor) \qquad (Eq.\ 5.5)$$

5.4.5 Cálculo dos Pontos de Caso de Uso Técnicos

Depois de calculados os valores de TUUCP (*Technical Unadjusted Use Case Points*), e TCF (*Technical Complexity Factor*), calcula-se a TUCP (*Technical Use Case Points*) ajustada para o sistema todo, conforme a *Eq*. 5.6.

$$TUCP = TUUCP * TCF \qquad (Eq.\ 5.6)$$

Ressalta-se que os fatores ambientais (EF) não são levados em consideração para efeito de cálculo na estimativa de tamanho da TUCP, que considera apenas fatores técnicos e não-técnicos. Entende-se que o tamanho do sistema seja independente de como o sistema deva ser construído, ou da influência que possa ter dos fatores ambientais, tais como experiência da equipe, motivação, familiaridade com o processo de desenvolvimento, linguagem de programação, experiência profissional.

Uma caracterização disto está na engenharia civil, onde na construção de uma parede, por exemplo, o tamanho está relacionado apenas com a extensão a ser construída e com a quantidade de tijolos em metros quadrados m^2 a ser utilizada para erguer o muro. A

experiência do pedreiro, a qualidade material e a técnica utilizada na construção não estão relacionadas diretamente com o tamanho do artefato a ser produzido.

A TUCP é uma métrica de estimativa direta, dentro da categoria de modelos paramétricos e com a abordagem *top-dowm*, conforme definido no capítulo 4. Assim como a UCP, a TUCP também estima o software como um produto único.

Pelo motivo de não se ter um detalhamento das estimativas de tamanho na UCP, este trabalho objetiva fornecer também uma granularidade menor nas estimativas de tamanho, para que seja possível realizar o acompanhamento das atividades do projeto por etapas do ciclo de vida.

5.5 Estimativa de Tamanho por Etapa do Ciclo de Vida

A proposta deste trabalho é que o sistema seja estimado ao longo das etapas do ciclo de vida, para se obter uma granularidade mais apurada de estimativa, tornando mais fácil o acompanhamento de um projeto de software e permitindo ações corretivas antes do final de cada etapa do ciclo de vida desse projeto.

Como o esforço de um projeto é diretamente proporcional a seu tamanho, então o tamanho de um caso de uso por etapa do ciclo de vida pode ser baseado no percentual de esforço empregado para seu desenvolvimento. Assim sendo, e baseado no trabalho de Meneses (2001), definiu-se a estimativa de tamanho do caso de uso por etapa do ciclo de vida na *Eq*.5.7.

$$\mathrm{TUCP}_{(\mathrm{UC\_Etapa})} = \left(\left(\frac{\mathrm{TUCP}}{\sum \mathrm{TUUCW}} \right) * \mathrm{TUUCW}_{(\mathrm{UC})} \right) * PE_{(\mathrm{Etapa})} \qquad (Eq.\ 5.7)$$

A $\mathrm{TUCP}_{(\mathrm{UC\_Etapa})}$ é o tamanho do caso de uso por etapa do ciclo de vida, sendo definido proporcionalmente a seu fator de complexidade e ao percentual de distribuição de esforço para a etapa do ciclo de vida considerada. O $\mathrm{TUUCW}_{\mathrm{UC}}$ é o peso do caso de uso que está sendo calculado. O valor de *PE* corresponde ao percentual de esforço estimado para a atividade (ou etapa) em mensuração.

O tamanho TUCP do caso de uso como um todo ao longo do projeto é mostrado na *Eq*. 5.8.

$$\mathrm{TUCP}_{(\mathrm{UC})} = \left(\frac{\mathrm{TUCP}}{\sum \mathrm{TUUCW}} \right) * \mathrm{TUUCW}_{(\mathrm{UC})} \qquad (Eq.\ 5.8)$$

Portanto, o tamanho TUCP de uma etapa do projeto é dado pela *Eq*. 5.9.

$$\text{TUCP}_{(\text{Etapa})} = \text{TUCP} * PE_{(\text{Etapa})} \qquad (Eq.\ 5.9)$$

O percentual de esforço para cada etapa deve ser obtido segundo as características do projeto, da equipe de desenvolvimento e da organização. Caso a organização não tenha uma base histórica contendo tais informações de esforço, estes dados podem ser obtidos através dos dados da indústria de software ou da literatura.

Neste trabalho, consideram-se apenas as seguintes etapas do ciclo de vida de software, por serem consideradas as mais representativas: (*i*) Requisitos (*Req*); (*ii*) Análise e Projeto (*A&P*); (*iii*) Codificação (*Cod*); e (*iv*) Teste (*Tst*).

5.6 Cálculo dos Fatores Ambientais

Os fatores ambientais (EF) são calculados de forma similar aos apresentados pela técnica UCP (seção 4.7.1.5).

Da mesma maneira que a técnica original da UCP, os fatores ambientais dizem respeito aos requisitos não-funcionais associados ao projeto, tais como experiência da equipe, estabilidade do projeto e motivação dos programadores (RIBU, 2001).

Da mesma forma que fator técnico (TCF), o fator ambiental (EF) utiliza-se do mesmo mecanismo de peso, onde cada peso deve ser multiplicado pelo valor de 0 a 5. A Tabela 5.4 mostra os fatores ambientais previstos e seus respectivos pesos. Para os fatores de F1 a F4, 0 (zero) significa nenhuma experiência no assunto, 5 significa excelência no assunto e 3 um conhecimento mediano sobre o assunto. Para o fator F6, 0 significa requisitos instáveis, 5 imutáveis e 3 poucas mudanças. Para o fator F7, 0 significa nenhum colaborador por meio período, 5 significa que todos os colaboradores trabalham meio período e 3 que alguns colaboradores trabalham meio período. Para o fator F8, 0 significa uma linguagem com pouca ou nenhuma dificuldade de programação, 3 significa com uma complexidade média e 5 com uma alta complexidade e dificuldade.

Tabela 5.4 - Fatores ambientais (Ribu, 2001)

Fator	Descrição	Peso
F1	Familiaridade com o RUP ou outro processo formal	**1,5**
F2	Experiência com a aplicação em desenvolvimento	**0,5**
F3	Experiência com orientação a objeto	**1**
F4	Capacidade de análise	**0,5**
F5	Motivação	**1**
F6	Requerimentos estáveis	**2**
F7	Colaboradores de meio período	**-1**
F8	Dificuldade na linguagem de programação	**-1**

O fator ambiental (EF) é calculado pela *Eq*. 5.10 abaixo, onde o EFator é o somatório dos produtos entre o peso e a nota atribuída de cada fator ambiental.

$$EF = 1,4 + (-0,03 * EFator) \qquad (Eq.\ 5.10)$$

5.7 Estimativa de Esforço do Projeto

Para obter a estimativa de esforço total para o projeto, a técnica TUCP utiliza o fator de produtividade (PROD), o fator de ambiente (EF) e o tamanho TUCP, segundo a *Eq*. 5.11. Neste caso, o fator de ambiente (EF) retorna ao cálculo do esforço total do projeto de forma semelhante à UCP.

$$Esforço = TUCP * EF * PROD \qquad (Eq.\ 5.11)$$

O fator de produtividade pode ser calibrado de acordo com a produtividade da equipe por tipos de atividade de um projeto. Esse fator pode ser obtido através de uma base histórica organizacional, onde podem ser armazenados dados, como, por exemplo, o fator de produtividade para cada etapa do ciclo de vida e características de projeto. Caso essas informações não existam para uma determinada organização, alguns estudos empíricos foram realizados na literatura, podendo estes dados ser utilizados a priori e depois ser refinados para a organização e para o tipo de projeto.

Com base na estimativa de tamanho, Karner (1993) propôs uma produtividade de 20 homens/hora (h.h) por UCP.

Schneider e Winters *appud* Ribu (2001), analisando a proposta de Karner (1993), propuseram uma produtividade de 20 h.h por UCP para projetos onde a equipe seja

considerada estável e experiente, e o valor de 28 h.h por UCP para projetos em que os requisitos não sejam estáveis e com uma equipe não muito experiente.

Spaks *appud* Anda *et al*. (2001) mostrou que o esforço pode variar entre 15 h.h e 30 h.h por UCP.

Os valores para a produtividade devem estar de acordo com as características de projetos semelhantes desenvolvidos na instituição obtidos a partir de um repositório contendo a base histórica da produtividade organizacional por tipo de projeto.

5.7.1 Estimativa de Esforço por Etapa do Ciclo de Vida

Na UCP (KARNER, 1993), o fator produtividade é único para todo o projeto. No entanto, constatou-se que o fator de produtividade pode variar dependendo da etapa do ciclo de vida do projeto. Isto porque a experiência da equipe de desenvolvimento pode variar de uma etapa para outra, já que, normalmente, as pessoas envolvidas em uma etapa não são as mesmas de uma outra etapa. Além disso, outros fatores podem influenciar na produtividade tais como: tipo de equipe, linguagem, experiência, utilização de *frameworks*, etc.

A calibragem para a produtividade para cada etapa do ciclo de vida deve ser obtida da base histórica organizacional. Assim, o esforço por etapa do ciclo de vida do projeto é calculado na *Eq*. 5.12.

$$\text{Esforço}_{(\text{Etapa})} = \text{TUCP}_{(\text{Etapa})} * \text{EF} * \text{PROD}_{(\text{Etapa})} \qquad (Eq.\ 5.12)$$

Em decorrência da *Eq. 5.12*, o esforço por etapa do ciclo de vida para o caso de uso($\text{Esforço}_{(\text{UC\_Etapa})}$) e esforço do caso de uso ($\text{Esforço}_{(\text{UC})}$) podem ser obtidos conforme equação abaixo:

$$\text{Esforço}_{(\text{UC\_Etapa})} = \text{TUCP}_{(\text{UC\_Etapa})} * \text{EF} * \text{PROD}_{(\text{Etapa})}$$

A seguir, serão apresentadas as estimativas de prazo e de custo.

5.8 Estimativas de Prazo e de Custo

As estimativas de prazo e custos podem ser realizadas a partir do esforço estimado com a técnica proposta, disponibilidade de recursos e restrições do projeto. Essa estimativa pode ser feita para o projeto como um todo ou para cada etapa do ciclo de vida através do esforço estimado conforme prevê a TUCP.

Depois de calculado o esforço, poder-se-á determinar o total do tempo (na unidade de tempo considerada), que será necessário ao projeto, multiplicando-se o valor da TUCP pelo tempo médio gasto, segundo a *Eq*. 5.13. Em geral, a unidade de tempo utilizada é horas.

$$\text{Prazo}_{(\text{unidade\_tempo})} = \text{TUCP} * \text{Tempo\_médio}_{(\text{TUCP})} \qquad (Eq.\ 5.13)$$

O uso de técnicas do tipo PERT (*Programme Evaluation Review Technique*) e CPM (*Critical Path Method*) pode auxiliar na determinação do prazo da finalização do projeto, levando em consideração caminhos críticos, alocação de recursos, seqüenciamento de atividades e dependência de tarefas.

Além do prazo, a estimava de custo pode ser também determinada utilizando-se as equações *Eq*. 5.14 ou *Eq*. 5.15. O custo é calculado na unidade monetária considerada (por exemplo, em Reais).

$$\text{Custo}_{(\text{unidade\_monetária})} = \text{TUCP} * \text{Valor}_{(\text{TUCP})} \qquad (Eq.\ 5.14)$$

$$\text{Custo}_{(\text{unidade\_monetária})} = \text{Esforço} * \text{Valor}_{(\text{unidade\_esforço})} \qquad (Eq.\ 5.15)$$

No entanto, esta maneira de calcular o custo apenas está relacionada com o custo de esforço relacionado aos casos de usos do projeto relacionados. Os gastos relacionados a gestão de projeto, gestão de configuração, atividades de qualidade (*Quality Assurance*) e despesas indiretas do projeto como aquisição ou aluguel de hardware e software, viagens com reuniões e testes no cliente, contas telefônicas (por exemplo: ligações de longas distâncias, vídeo-conferência, linhas dedicadas para testes, etc.), treinamentos, despesas com os postos de trabalho, entre outros custos não estão sendo considerados.

Na seção abaixo, é mostrado um exemplo de aplicação da técnica TCUP.

5.9 Aplicação da TUCP

Para exemplificar a técnica TUCP, foi utilizado o Sistema de *Call Center* e um caso de uso desse sistema, “Manter Atendente” (UC_01), que foi especificado no Anexo C.

O Sistema de *Call Center* tem como propósito acompanhar a abertura de chamados de clientes, junto ao setor de assistência técnica de uma empresa. Este sistema é responsável por acompanhar o registro de um chamado, desde seu início até seu fechamento. Ele deverá ser desenvolvido na plataforma J2EE, ser portável e de fácil instalação. O tempo de resposta nas consultas deve ser considerado crítico para o sistema. O processo de contagem de TUCP passa pelas seguintes etapas:

Contagem dos atores (UAW)

A partir do levantamento dos requisitos do Sistema de *Call Center*, constatou-se que esse sistema possui quatro atores, sendo um simples e os outros três complexos. Os quatro atores do sistema são: usuário, administrador, atendente, e coordenador do *Call Center*. O administrador, o atendente e o coordenador do *Call Center* são atores complexos, e o usuário é um ator simples. O valor dos pesos do atores (UAW) é calculado conforme a Tabela 5.5.

Tabela 5.5 - Peso dos atores do sistema *Call Center*

Atores	Quantidade	Peso	Peso Total
Simples	1	1	(1 * 1) = 1
Complexo	3	3	(4 * 3) = 12
TOTAL			**13**

Neste caso, o valor do peso dos atores é: UAW = 13.

Contagem dos casos de uso (TUUCW)

Este sistema contém apenas oito casos de uso, onde dois são simples, quatro são intermediários, um complexo, e um *n*-complexo, segundo a notação da TUCP. O peso dos casos de uso (TUUCW) para o sistema de *Call Center* é calculado conforme a Tabela 5.6.

Tabela 5.6 - Peso dos casos de uso do sistema de *Call Center*

Complexidade	Nº Transações	Quantidade	Peso (TUUCW)	Peso Total
Simples	1 - 3T	2	5	(2 * 5) = 10
Médio	4 - 7T	4	10	(4 * 10) = 40
Complexo	8 - 11T	1	15	(1 * 15) = 15
N- Complexo	13T	1	20	(1 * 20) = 20
TOTAL				**85**

Para o cálculo do peso do caso de uso *n*-complexo com 13 transações (T = 13), foi considerado k = 4 (veja seção 5.4.2). Assim, utilizando-se a *Eq*. 5.3, t = 11. Pela *Eq*. 5.2, $n = T / t = 13 / 11$. Como $r \in [1, 3]$, então p = 5. Assim, pela *Eq*. 5.1, tem-se o valor de TUUCW para este caso de uso:

$$\text{TUUCW} = 15 \text{ x } 1 + 5 = 20$$

Contagem dos pontos de casos de uso não ajustados (TUUCP)

O valor de TUUPC é calculado a seguir de acordo com a *Eq*. 5.4.

$$\text{TUUCP} = \sum \text{UAW} + \sum \text{TUUCW} = 13 + 85 = 98$$

Cálculo dos fatores de complexidade técnica (TCF)

Os fatores de complexidade técnica (TCF) devem ser calculados de acordo com as características não funcionais do sistema, conforme a Tabela 5.7.

Tabela 5.7 - Fatores de complexidade técnica para o sistema *Call Center*

Fator	Descrição	Peso	Valor	Peso Total
T1	O sistema é distribuído	2,0	2	4
T2	O sistema deve ser otimizado	1,0	5	5
T3	A eficiência do Usuário-Final não é extremamente necessária nesse tipo de aplicação	1,0	5	5
T4	A aplicação não possui processamento interno complexo	1,0	2	2
T5	Código deve ser reutilizável	1,0	2	2
T6	Fácil para o usuário final instalar	0,5	2	1
T7	Fácil para o usuário final usar	0,5	5	2,5
T8	É portável	2,0	2	4
T9	É fácil de manter	1,0	2	2
T10	Por ser ambiente Web, há necessidade de controle de concorrência eficiente	1,0	3	3
T11	Necessidade de autenticação de usuário	1,0	3	3
T12	Não há acesso direto para terceiros	1,0	3	3
T13	Não há necessidade de treinamento, por ser bastante simples e intuitivo	1,0	0	0
TOTAL				**36,5**

O cálculo do TFator é dado por: TFator = ∑Fator

= (2*2) + (5*1) + (5*1) + (2*1) + (2*1) + (2*0,5) + (5*0,5) + (2*2) + (2*1) + (3*1) + (3*1) + (3*1) + (0*1)

$$TFator = 36,5$$

A partir da *Eq*. 5.5, calcula-se o valor de TCF:

TCF = 0,6 + (0,01 * 36,5)

$$TCF = 0,965$$

Cálculo dos pontos de caso de uso técnicos (TUCP)

Da *Eq*. 5.6, chega-se, finalmente, ao cálculo do tamanho do sistema, isto é, do cálculo dos pontos de caso de uso técnicos (TUCP):

$$TUCP = 98 * 0,965 = 94,57$$

A Tabela 5.8 mostra os oito casos de uso do sistema de *Call Center* com seus valores de TUUCW e TUCP.

Tabela 5.8 - Peso dos casos de uso do sistema de *Call Center*

Caso de Uso	(TUUCW)	TUCP
UC_01	**20**	**22,251**
UC_02	5	5,562
UC_03	**5**	**5,562**
UC_04	**10**	**11,125**
UC_05	10	11,125
UC_06	10	11,125
UC_07	10	11,125
UC_08	15	16,688
TOTAL	**85**	**94,57**

Estimativa de Tamanho por Etapa do Ciclo de Vida

O valor encontrado pela TUCP corresponde ao tamanho total do sistema. Para se obter a estimativa de tamanho na etapa de requisitos do caso de uso UC_01, deve-se aplicar a *Eq*. 5.7.

O *percentual de esforço* para especificar este caso de uso UC_01 foi estimado em aproximadamente 21% (valor obtido nos ativos da organização).

$$\text{TUCP}_{(\text{UC\_01\_Req})} = \left(\left(\frac{94,57}{85}\right) * \ 20\right) * 0,21 = 4,67 \ .$$

Então, a estimativa de tamanho para a etapa de requisitos do caso de uso UC_01 é de 4,89.

Cálculo dos Fatores Ambientais

Os fatores ambientais (EF) para o sistema de *Call Center* estão calculados conforme a Tabela 5.9

Tabela 5.9 – Fatores ambientais para o sistema *Call Center*

Fator	Descrição	Peso	Valor	Peso Total
F1	Familiaridade com o RUP	1,5	5	7,5
F2	Experiência em desenvolvimento	0,5	3	1,5
F3	Experiência em OO	1	5	5
F4	Capacidade de análise	0,5	4	2
F5	Motivação	1	3	3
F6	Pode haver mudanças no negócio	2	3	6
F7	Colaboradores em meio período	- 1	0	0
F8	Dificuldade na linguagem de programação	- 1	3	- 3
TOTAL				**22**

O cálculo do TFator é dado por: TFator = ∑Fator

EFator = (5*1,5) + (3*0,5) + (5*1) + (4*0,5) + (3*1) + (3*2) + (0*-1) + (3*-1)

TFator = 22

A partir da *Eq*. 5.10, calcula-se o valor de EF:

EF = 1,4 + (-0,03 * 22)

EF = 0,74

Estimativa de Esforço do Projeto

A produtividade utilizada para o cálculo do esforço do Sistema de *Call Center* foi estimada em 20 h.h (PROD = 20), pois a equipe alocada para desenvolvê-lo apresenta um perfil técnico alto, e experiências no negócio e na linguagem de programação. Assim, o esforço deste sistema pode ser calculado pela *Eq*. 5.11.

$$\text{Esforço} = 94{,}57 * 0{,}74 * 20 = 1.399{,}63 \text{ h.h}.$$

Estimativa de Esforço por Etapa do Ciclo de Vida

A estimativa de esforço na etapa de requisitos deste caso de uso (UC_01) é calculada pela *Eq*. 5.12 e apresentada abaixo. A produtividade utilizada neste caso foi estimada em 10 h.h, uma vez que a equipe de requisitos é muito experiente.

$$\text{Esforço}_{(\text{UC\_01\_Req})} = 1{,}164 * 0{,}74 * 10 = 8{,}61 \text{h.h} \ .$$

Estimativas de Prazo e de Custo

Para se estimar o prazo (em horas) necessário ao desenvolvimento do Sistema de *Call Center*, utiliza-se a *Eq*. 5.13, como apresentado abaixo. O valor médio de horas por TUCP utilizado será de 16 horas (valor obtido historicamente na organização).

$$\text{Prazo}_{(\text{horas})} = 94{,}57 * 16 = 1.545{,}12 \text{h}.$$

O custo do Sistema de *Call Center* pode ser obtido através da *Eq*. 5.14. Foi considerado o valor por TUCP de R$ 30,00. Portanto, para esse sistema, o custo em relação ao desenvolvimento do projeto, não levando em consideração as despesas indiretas do projeto, é calculado como:

$$\text{Custo}_{(\text{R\$})} = 94{,}57 * 30{,}00 = \text{R\$ } 2.837{,}10 \ .$$

A seguir, será apresentada uma proposta para o acompanhamento do progresso funcional ao longo do desenvolvimento de software.

5.10 Progresso Funcional

Com o objetivo de auxiliar no acompanhamento e monitoramento do sistema, este trabalho propõe uma visão de progresso funcional ao longo do ciclo de vida do sistema (MENESES, 2001; MENESES, 2000a; MENESES, 2000b), a partir a aplicação da técnica TUCP. Neste contexto, é mensurado o percentual da funcionalidade de todo o sistema, a partir da análise do progresso de cada caso de uso do projeto.

A visão funcional tem como informação essencial a métrica global, $\mu_{Sistema}$, que indica quantitativamente o progresso funcional do sistema. Essa métrica permitirá, entre outros aspectos, visualizar o progresso da funcionalidade atual, comparar tal crescimento com incrementos e fases anteriores, identificar possíveis dificuldades existentes no sistema, e evidenciar casos de uso mal projetados (MENESES, 2001).

A métrica $\mu_{Sistema}$ é derivada da *Eq*. 5.16, definida por Champeaux (1997) *appud* Meneses (2001), que corresponde ao somatório da contribuição de cada artefato que compõe a métrica μ_x. O valor de *i* representa o conjunto de artefatos relevantes de *x*, e D_i representa a contribuição do artefato *i* para a métrica μ_x.

$$\mu_x = \sum_i D_i \qquad (Eq.\ 5.16)$$

Para o acompanhamento e o monitoramento mais adequado do sistema, pode-se fazer uso das seguintes visões: (*i*) progresso funcional do sistema; (*ii*) progresso funcional da etapa do ciclo de vida; (*iii*) progresso funcional do caso de uso.

Progresso Funcional do Sistema

O progresso funcional do sistema ($\mu_{Sistema}$) é um valor percentual que se deriva diretamente do progresso funcional de cada caso de uso (μ_{UC}) do próprio sistema, conforme a *Eq*. 5.17.

$$\mu_{Sistema} = \sum \mu_{UC} \qquad (Eq.\ 5.17)$$

Como os casos de uso freqüentemente apresentam complexidade variada (TUCP) e necessitam de níveis de esforço distintos para sua realização ao longo das etapas do ciclo de vida, pode-se obter também o progresso funcional do sistema através da *Eq*. 5.18. O μ_{Etapa} indica o progresso do caso de uso de uma etapa do ciclo de vida em valor percentual.

$$\mu_{Sistema} = \frac{\sum \left(TUCP_{etapa} * \mu_{etapa} \right)}{\sum TUCP_{etapa}} \qquad (Eq.\ 5.18)$$

Progresso Funcional da Etapa do Ciclo de Vida

A métrica relacionada ao progresso funcional da etapa do ciclo de vida do sistema pode ser calculada através da *Eq*. 5.19. Neste caso, μ_{UC_Etapa} é um valor percentual que indica o progresso funcional do caso de uso em uma dada etapa do ciclo de vida. Esse progresso pode ser obtido em função da elaboração dos produtos de trabalho naquela etapa, isto é, o quanto cada produto de trabalho já foi realizado.

$$\mu_{etapa} = \frac{\sum \left(TUCP_{UC\_Etapa} * \mu_{UC\_Etapa} \right)}{\sum TUCP_{UC\_Etapa}} \qquad (Eq\ 5.19)$$

Progresso Funcional do Caso de Uso

A métrica relacionada ao progresso funcional de um caso de uso pode ser calculada através da *Eq*. 5.20. Assim, μ_{UC} é um valor percentual que indica o progresso funcional do caso de uso, considerando-se suas etapa do ciclo de vida. Neste trabalho são consideradas as seguintes etapas (seção 5.5): *Req*, *A&P*, *Cod*, e *Tst*.

$$\mu_{UC} = \frac{\sum_{Etapa \in \{Req, A\&P, Cod, Tst\}} \left(TUCP_{UC\_Etapa} * \mu_{UC\_Etapa} \right)}{\sum_{Etapa \in \{Req, A\&P, Cod, Tst\}} TUCP_{UC\_Etapa}} \qquad (Eq.\ 5.20)$$

5.11 Aplicação do Progresso Funcional

Para exemplificar o progresso funcional, foi utilizado também o Sistema de *Call Center*. Para os casos de uso da Tabela 5.8. foram considerados os seguintes *percentuais de esforço* por etapa do ciclo de vida (em valores decimais), segundo a Tabela 5.10.

Tabela 5.10 – Percentual de esforço (*PE*) por etapa de ciclo de vida

Caso de Uso	PE_{Req}	$PE_{A\&P}$	PE_{Cod}	PE_{Tst}
UC_01	**0,21**	**0,26**	**0,43**	**0,10**
UC_02	0,19	0,25	0,50	0,06
UC_03	**0,19**	**0,25**	**0,50**	**0,06**
UC_04	**0,17**	**0,33**	**0,33**	**0,17**
UC_05	0,17	0,33	0,33	0,17
UC_06	0,17	0,33	0,33	0,17
UC_07	0,17	0,33	0,33	0,17
UC_08	0,20	0,30	0,40	0,10

A partir da Tabela 5.8 e da Tabela 5.10, é calculada a estimativa de tamanho por etapa do ciclo de vida, conforme a Tabela 5.11.

Tabela 5.11 – TUCP por etapa de ciclo de vida por caso de uso

Caso de Uso	$TUCP_{Req}$	$TUCP_{A\&P}$	$TUCP_{Cod}$	$TUCP_{Tst}$	$TUCP_{UC}$
UC_01	**4,673**	**5,785**	**9,568**	**2,225**	**22,25**
UC_02	1,057	1,391	2,781	0,334	**5,56**
UC_03	**1,057**	**1,391**	**2,781**	**0,334**	**5,56**
UC_04	**1,891**	**3,672**	**3,672**	**1,891**	**11,13**
UC_05	1,891	3,672	3,672	1,891	**11,13**
UC_06	1,891	3,672	3,672	1,891	**11,13**
UC_07	1,891	3,672	3,672	1,891	**11,13**
UC_08	3,338	5,007	6,676	1,669	**16,69**
TOTAL	**17,689**	**28,260**	**36,493**	**12,127**	**94,57**

O próximo passo é obter o μ_{Etapa}, que indica o progresso de cada caso de uso nas etapas do ciclo de vida, em valor percentual. Para o Sistema de *Call Center* será considerado que apenas os casos de uso UC_01, UC_03, e UC_04 foram iniciados; os demais casos de uso deverão ser iniciados na próxima interação. Serão atribuídos valores percentuais do subconjunto {0,00; 0,25; 0,50; 0,75; 1,00} para indicar o progresso funcional de cada um desses três casos de uso por etapa do ciclo de vida (Tabela 5.12).

Tabela 5.12 – Progresso funcional de caso de uso por etapa de ciclo de vida

Caso de Uso	μ_{Req}	$\mu_{A\&P}$	μ_{Cod}	μ_{Tst}	μ_{UC}
UC_01	1,00	1,00	1,00	1,00	1,00
UC_03	1,00	1,00	0,75	0,50	0,84
UC_04	0,50	0,75	0,00	0,00	0,33

Portanto, a partir da Tabela 5.11 e da Tabela 5.12 é calculado abaixo o progresso funcional dos casos de uso UC_01, UC_03, e UC_04 através da *Eq*. 5.20.

$$\mu_{UC\_01} = \frac{4{,}673*1{,}00 + 5{,}785*1{,}00 + 9{,}568*1{,}00 \; + \; 2{,}225*1{,}00}{22{,}25} = 1{,}00$$

$$\mu_{UC\_03} = \frac{1{,}057*1{,}00 + 1{,}391*1{,}00 + 2{,}781*0{,}75 + 0{,}334*0{,}50}{5{,}56} = 0{,}84$$

$$\mu_{UC\_04} = \frac{1{,}891*0{,}50 + 3{,}672*0{,}75 + 3{,}8672*0{,}00 + 1{,}1891*0{,}00}{11{,}13} = 0{,}33$$

Pelos dados acima, em relação ao que foi estimado para o progresso funcional dos casos de uso, o caso de uso UC_01 já foi concluído; o caso de uso UC_03 foi 84% realizado; e o caso de uso UC_04 foi 33% realizado.

O progresso funcional de cada fase do ciclo de vida do sistema é calculado abaixo através da *Eq*. 5.19.

$$\mu_{Req} = \frac{4{,}673*1{,}00 + 1{,}057*1{,}00 + 1{,}891*0{,}50}{17{,}689} = 0{,}377$$

$$\mu_{A\&P} = \frac{5{,}785*1{,}00 + 1{,}391*1{,}00 + 3{,}672*0{,}75}{28{,}260} = 0{,}351$$

$$\mu_{Cod} = \frac{9{,}568*1{,}00 + 2{,}781*1{,}00 + 3{,}672*0{,}50}{36{,}493} = 0{,}388$$

$$\mu_{Tst} = \frac{2{,}225*1{,}00 + 0{,}334*0{,}50 + 1{,}891*0{,}00}{12{,}127} = 0{,}197$$

Pelos dados acima, em relação ao que foi estimado do progresso funcional por etapa do ciclo de vida, a etapa de requisitos (*Req*) foi realizada em 37,7%; a etapa de Análise e Projeto (*A&P*) em 35,1%; a etapa de Codificação (*Cod*) em 38,8%; e a etapa de Teste (*Tst*) em 19,7%.

O cálculo do progresso funcional do sistema de *Call Center* ($\mu_{Sistema}$) é obtido abaixo através da *Eq*. 5.18.

$$\mu_{Call\ Center} = \frac{17{,}689*0{,}377 + 28{,}260*0{,}351 + 36{,}493*0{,}388 + 12{,}127*0{,}197}{94{,}57} = 0{,}350$$

Pelos dados acima, em relação ao que foi estimado, o progresso funcional do sistema de *Call Center* foi de 35,0%.

5.12 Conclusão

Este capítulo delineou em detalhes a proposta da técnica TUCP como uma extensão da técnica UCP (KARNER, 1993).

O próximo capítulo apresentará os resultados da aplicação da TUCP em alguns projetos de software em uma empresa certificada CMM, nível 2.

Capítulo 6

Estudo de Casos

Neste capítulo, serão apresentados experimentos em estimativas de projetos de software, que foram realizados em uma empresa de software, objetivando validar a técnica TUCP, através da investigação dos resultados obtidos.

Este capítulo apresenta os resultados obtidos durante a aplicação da técnica TUCP (*Technical Use Case Points*) proposta, que é uma extensão da técnica UCP (*Use Case Points*), em vários projetos reais de software, na empresa de software denominada Instituto Atlântico. A realização desses experimentos objetivou validar a nova proposta, buscando avaliar a efetividade dos resultados obtidos com a sua utilização. Este estudo de caso apresenta também a realização do acompanhamento de tamanho das etapas do ciclo de vida de um projeto de software, baseado no tamanho dos casos de uso, permitindo assim aos gerentes de projeto realizarem uma avaliação do progresso funcional do projeto de acordo com a complexidade do sistema.

A seguir, será apresentado o perfil da organização, onde os experimentos foram realizados e, em seguida, serão expostos os resultados obtidos dos projetos selecionados para compor este estudo de casos.

6.1 Perfil da Organização

O Instituto Atlântico é uma instituição de pesquisa e desenvolvimento (P&D), fundada em novembro de 2001. É uma organização certificada como SW-CMM, nível 2, desde outubro de 2003. Tem com meta organizacional ser certificada CMMI, nível 3, até dezembro de 2005.

Sua missão é desenvolver e difundir tecnologias de alto valor agregado para o sucesso de seus clientes, através de relações duradouras com parceiros que resultem em benefícios para a sociedade. Baseado nos valores de inovação, excelência, comprometimento e respeito, o Instituto Atlântico busca a melhoria contínua da qualidade de seus processos de software,

visando o aumento de produtividade, o cumprimento de prazos estabelecidos, a redução de custos do produto final e, sobretudo, atender cada vez melhor a seus clientes.

O escopo de atuação do Instituto abrange pesquisa e desenvolvimento de soluções de software (Web e Automação de processos), hardware, e serviços de consultoria na área de tecnologia da informação e telecomunicações. Essas soluções são desenvolvidas mediante projetos próprios ou específicos para clientes, e podem ter como objetivo uma nova tecnologia, um novo produto ou a atualização de algum produto já existente. Os projetos desenvolvidos pertencem a diversas áreas, possuem tamanhos variados e são desenvolvidos em diferentes plataformas (J2EE, J2ME, .NET, C, C++, etc.).

A dinâmica de trabalho do Instituto Atlântico, na condução de projetos, é em geral operacionalizada como se segue: cada projeto tem um *gerente de projeto*, um *coordenador de projeto*, *analistas de sistemas*, e *programadores*, conforme Figura 6.1.

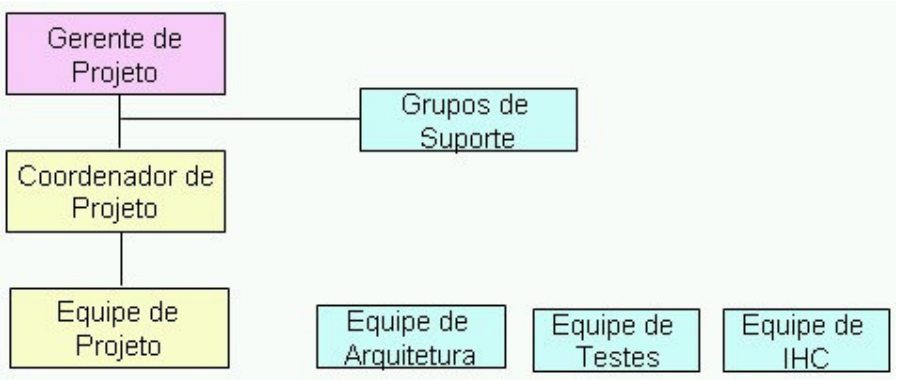


Figura 6.1 - Estrutura de trabalho do Instituto Atlântico

O *gerente de projeto* tem como responsabilidades fazer o acompanhamento gerencial técnico, financeiro e dos recursos humanos para o projeto, negociar alterações de compromissos com o cliente, quando envolver mudanças contratuais, além de designar o coordenador de projeto.

O *coordenador de projeto* é responsável por coordenar tecnicamente a equipe de projeto, planejar e acompanhar o projeto, conduzir reuniões de acompanhamento técnico, tomar as medidas pertinentes para corrigir possíveis desvios encontrados e acompanhar a execução das correções até a sua solução efetiva, alocar pessoal para os papéis de líder de requisitos e líder de configuração, entre outros papéis, negociar compromissos com o Grupos de Suporte: Equipe de Arquitetura, Equipe de Testes, e Equipe de IHC (Interface Humano-Computador), além de negociar a alocação de recursos humanos e financeiros com o gerente de projeto.

A *equipe de projeto* é composta por analistas de sistemas e programadores, havendo dois papéis fundamentais no processo desta organização: o *líder de requisitos* e o *líder de configuração*.

O *líder de requisitos* tem como atividades especificar requisitos funcionais, especificar requisitos não-funcionais, detalhar casos de uso, gerenciar requisitos, além de poder executar outras atividades no projeto (analisar, projetar, codificar e testar, dependendo de seu perfil).

O *líder de configuração* tem a responsabilidade de planejar as atividades de gestão de configuração, fechar as *baselines*[1], auditar as *baselines*, executar as atividades planejadas de gerência de configuração, além de poder executar atividades operacionais do projeto.

Apesar de a equipe possuir competências para desenvolver as atividades necessárias no projeto, ainda conta com o apoio do *Grupo de Suporte* institucional, que é composto pela *equipe de arquitetura*, *equipe de testes*, e *equipe de IHC*. A *equipe de arquitetura* tem a responsabilidade de participar direta ou indiretamente (consultoria) da elaboração e da revisão da arquitetura dos projetos, apoiar o líder de requisitos nas especificações de requisitos do projeto, quando necessário. A *equipe de testes* é responsável por executar os testes sistêmicos[2] do projeto, gerar relatórios com resultados desses testes, além de apoiar grupos internos do projeto em testes (unitários e integração, por exemplo).

A responsabilidade de executar os teste unitários e de integração é da equipe de projeto e não da equipe de testes. Isto porque os testes unitários são o estágio mais baixo de escala de testes, que são aplicados aos menores componentes de código criados, visando garantir que estes atendam às especificações, em termos de características e de funcionalidades. Os testes unitários verificam o funcionamento de uma parte do sistema isoladamente ou de partes que possam ser testadas separadamente, podendo, inclusive, ser um programa ou um componente (MOREIRA *et a*l, 2003).

Já os testes de integração são executados em uma combinação de componentes, para verificar se funcionam corretamente juntos, ou seja, para assegurar que as interfaces funcionem adequadamente e que os dados sejam processados de forma precisa, conforme as especificações. Componentes podem ser partes de código, módulos, aplicações distintas,

[1] Conjunto revisto e aprovado formalmente de resultados de um projeto, associado a um marco deste projeto, que serve de base para o desenvolvimento futuro, e que só pode ser alterado através de procedimento formalizado (Pádua, 2003).

[2] Os testes sistêmicos são realizados pela equipe de testes, visando à execução do sistema como um todo ou um subsistema (parte do sistema), dentro de um ambiente operacional controlado, para validar a exatidão e a perfeição na execução de suas funções. Neste estágio de testes, deve ser simulada a operação normal do sistema, sendo testadas todas as suas funções da forma mais próxima possível do que irá acontecer no ambiente de produção. Neste contexto, devem ser realizados os testes de carga, performance, usabilidade, compatibilidade, segurança e recuperação (MOREIRA *et al*, 2003).

clientes, servidores, etc. Por estas características, os testes unitários e de integração devem são realizados pela equipe de projeto.

Apesar da segmentação de papéis que existe na organização, um membro da equipe do projeto, quando se justificar, poderá vir a desempenhar papéis pertencentes ao grupo de suporte.

Na seção seguinte, serão apresentados os experimentos realizados em cinco projetos de software, desenvolvidos no Instituto Atlântico, que utilizaram a técnica de estimativas TUCP ao longo de seu ciclo de vida. O ciclo de vida utilizado foi o iterativo- incremental. Esses projetos, por questão de sigilo de suas informações, serão denominados de Projeto A, Projeto B, Projeto C, Projeto D, e Projeto E.

Para estes cinco projetos, são apresentados na Tabela 6.1 os *percentuais de esforço* por etapa de ciclo de vida, segundo a média histórica do Instituto. Dependendo das características de cada projeto, alterações nesses percentuais podem ser realizadas durante as estimativas. No entanto, o coordenador do projeto deverá ter uma *expertise* com relação a esta tomada de decisão.

Tabela 6.1 - Percentual de esforço por etapa do ciclo de vida

Etapa do Ciclo de Vida	Percentual de Esforço
Requisitos (*Req*)	20%
Análise e Projeto (*A&P*)	25%
Codificação (*Cod*)	45%
Testes (*Tst*)	10%

Para todos os projetos avaliados, no que se refere ao cálculo do *percentual de erro estimado* do esforço por etapa do ciclo de vida e para todo o projeto, foi utilizada a métrica *Symmetric Relative Error* (SER), proposta por M. Jorgensen e D. Sjobeg *appud* Ribu (2001), onde:

$$\text{SER} = (\text{Real} - \text{Estimado}) / \text{Real} \Leftrightarrow \text{Real} \leq \text{Estimado}$$
$$\text{SER} = (\text{Real} - \text{Estimado}) / \text{Estimado} \Leftrightarrow \text{Real} \geq \text{Estimado}$$

Vale salientar que o Instituto passou a utilizar efetivamente a TUCP no cálculo das estimativas de seus projetos de software desde o início de 2004, depois de ter tido problemas de estimativas (subestimadas), com o emprego da UCP em projetos de grande porte, especialmente aqueles que apresentavam mais de 12 transações por caso de uso. Atualmente, mais de 60 projetos já utilizaram a TUCP e em todos eles os resultados das estimativas foram

mais precisos que a UPC original. Apenas em alguns projetos, onde não havia nenhum caso de uso n-complexo, a precisão entre as duas técnicas foi a mesma.

6.2 Projeto A

O *Projeto A* foi desenvolvido em uma interface Web com J2EE, trabalhando com componentes ao longo de seu processo de desenvolvimento.

A equipe deste projeto era composta de seis membros, sendo que o coordenador do projeto foi alocado apenas para as atividades de planejamento e acompanhamento. Assim sendo, o projeto contou com cinco membros para as atividades operacionais (levantamento de requisitos, análise e projeto, codificação e testes). Um desses membros desempenhou o papel de *líder de requisitos* e um outro de *líder de configuração*. Esses líderes tinham alocado 50% de seu tempo para atividades específicas de seus papéis, e o restante do tempo para atuar como analista de sistema, ou outras atividades operacionais do projeto. O restante dos membros da equipe era composto de desenvolvedores.

À equipe do projeto foi alocado também um testador, pertencente ao Grupo de Suporte, que foi responsável por realizar os testes sistêmicos. Foi alocado também um arquiteto de software, que atuou somente como consultor e revisor da arquitetura sugerida pela equipe de projeto. A equipe de IHC desenvolveu as interfaces gráficas para este projeto.

A equipe do projeto era considerada experiente, pois já havia desenvolvido um produto de software com características semelhantes, além de ter um bom entrosamento, pois seus membros já haviam trabalhado juntos em outros projetos. O coordenador de projeto era também experiente, por ter gerenciado outros projetos com a mesma equipe e ter obtido êxito.

No decorrer do ciclo de vida deste projeto, merecem destaque os seguintes problemas que surgiram e que a equipe do projeto teve de tratá-los adequadamente:

i) utilização de um *framework* de desenvolvimento não conhecido pelos membros da equipe, que apresentou alguns *bugs* em sua utilização. Esses *bugs* foram corrigidos pela equipe do projeto;
ii) saída de dois membros experientes da equipe alocados em tempo integral e a posterior alocação de outros três membros à equipe do projeto, sendo um em tempo integral e os outros dois em tempo parcial, o que trouxe alguns problemas de entrosamento e de produtividade no decorrer do projeto;
iii) dificuldades de relacionamento com cliente e usuários deste projeto; e
iv) quantidade de testes sobre os artefatos muito superior ao previsto; muitos artefatos tiveram que ser testados novamente, por exigências do próprio cliente.

A seguir serão apresentados os dados do *Projeto A* utilizando a técnica TUCP proposta neste trabalho.

Fatores de Complexidade Técnica

Os fatores de complexidade técnica (TCF) (seção 5.4.4) relacionados ao *Projeto A* estão descritos na Tabela 6.2.

Tabela 6.2 - Fatores de complexidade técnica do *Projeto A*

Fator	Descrição	Peso	Valor	Ponderação
T1	Sistema distribuído	2,0	5	10,0
T2	Questões de desempenho de resposta (*throughputi*)	1,0	5	5,0
T3	Eficiência de usuário final (*online*)	1,0	2	2,0
T4	Processamento interno complexo	1,0	4	4,0
T5	Código deve ser reutilizável	1,0	4	4,0
T6	Fácil de instalar	0,5	2	1,0
T7	Fácil de usar	0,5	3	1,5
T8	Portável	2,0	3	6,0
T9	Fácil de modificar	1,0	4	4,0
T10	Concorrente	1,0	5	5,0
T11	Características especiais de segurança	1,0	4	4,0
T12	Acesso direto a componentes de terceiros	1,0	3	3,0
T13	Facilidades especiais de treinamento de usuários requeridas	1,0	3	3,0
TFator				**52,5**

O valor da coluna "Ponderação" é o produto entre o valor do peso de cada fator técnico com o valor atribuído para cada fator de complexidade de acordo com a característica do projeto. O cálculo do TCF (*Eq*. 5.5) é mostrado abaixo:

$$TCF = 0,6 + (0,01 * 52,5) = 1,125 .$$

Fatores de Complexidade Ambiental

Os fatores de complexidade ambiental (EF) dizem respeito aos requisitos não-funcionais associados ao processo de desenvolvimento, tais como experiência da equipe, estabilidade do projeto, e motivação dos programadores (seção 5.6). Conforme a Tabela 6.3, a equipe deste projeto pode ser considerada experiente.

Tabela 6.3 - Fatores de complexidade ambiental do *Projeto A*

Fator	Descrição	Peso	Valor	Ponderação
F1	Familiar com o Processo do Institucional	1,5	4	6,0
F2	Experiência com a aplicação	0,5	3	1,5
F3	Experiência em orientação a objetos	1,0	4	4,0
F4	Liderança Técnica	0,5	4	2,0
F5	Motivação	1,0	4	4,0
F6	Requisitos estáveis	2,0	3	6,0
F7	Integrantes em tempo parcial	-1,0	2	-2,0
F8	Linguagem de programação complexa	-1,0	2	-2,0
EFactor				**19,5**

O valor do fator de complexidade ambiental (EF) para o *Projeto A* foi calculado pela equação *Eq*. 5.10 abaixo.

$$EF = 1{,}4 + (-0{,}03 * 19{,}5) \cong 0{,}815$$

Cálculo da TUCP

O *Projeto A* desenvolveu 24 casos de usos, onde: sete eram simples, dez eram intermediários, um era complexo, e seis eram *n*-complexos. A Tabela 6.4 apresenta esses casos de uso, juntamente com o cálculo de UUCW (da técnica UCP), o cálculo de TUUCW, da TUCP (seção 5.4), e cálculo de $TUCP_{(UC)}$ (seção 5.5).

Tabela 6.4 - Ponderação de casos de uso por transações

Caso de Uso	Transações	Nº Transações	Peso (UUCW)	Peso (TUUCW)	TUCP
UC_01	simples	3T	5	5	5,7
UC_02	intermediário	4T	10	10	11,4
UC_03	simples	3T	5	5	5,7
UC_04	simples	3T	5	5	5,7
UC_05	intermediário	5T	10	10	11,4
UC_06	intermediário	4T	10	10	11,4
UC_07	intermediário	4T	10	10	11,4
UC_08	simples	3T	5	5	5,7
UC_09	intermediário	5T	10	10	11,4
UC_10	***n*-complexo**	**20T**	**15**	**30**	**34,1**
UC_11	***n*-complexo**	**18T**	**15**	**25**	**28,4**
UC_12	***n*-complexo**	**19T**	**15**	**30**	**34,1**
UC_13	intermediário	5T	10	10	11,4
UC_14	***n*-complexo**	**16T**	**15**	**25**	**28,4**
UC_15	complexo	9T	15	15	17,0
UC_16	simples	2T	5	5	5,7
UC_17	intermediário	7T	10	10	11,4
UC_18	simples	2T	5	5	5,7
UC_19	***n*-complexo**	**17T**	**15**	**25**	**28,4**
UC_20	intermediário	4T	10	10	11,4
UC_21	***n*-complexo**	**18T**	**15**	**25**	**28,4**
UC_22	intermediário	6T	10	10	11,4
UC_23	simples	3T	5	5	5,7
UC_24	intermediário	4T	10	10	11,4
TOTAL		**184T**	**240**	**310**	**352,0**

O valor calculado para o somatório dos pesos dos atores foi igual a $\sum UAW = 3$.

Como $TUUCP = \sum UAW + \sum TUUCW$, então $TUUCP = 3 + 310 = 313$.

Portanto, a $TUCP = 313 * 1{,}125 \cong 352{,}1$.

Para o *Projeto A*, a partir de dados retirados dos ativos organizacionais, foram considerados os *percentuais de esforço* por etapa do ciclo de vida da Tabela 6.1.

Na Tabela 6.5, serão apresentados os valores dos tamanhos em TUCP, e do esforço em h.h (homens-hora) (a partir do *percentual de esforço* acima), para cada caso de uso, e por etapa de ciclo de vida (seção 5.5): Requisitos (REQ), Análise e Projeto (A&P), Codificação (COD), e Teste (TST).

Neste projeto, serão detalhados apenas os cálculos do caso de uso UC_01, para exemplificar a técnica TUCP. Para calcular o tamanho desse caso de uso foi aplicada a *Eq*. 5.8 abaixo. Assim sendo, o tamanho do caso de uso UC_01 é 5,7.

$$\text{TUCP}_{(\text{UC\_01})} = \left(\frac{352{,}1}{310} \right) * 5 \cong 5{,}7$$

Tabela 6.5 - Distribuição de tamanho (TUCP) e esforço (h.h.) por etapa do ciclo de vida

Caso de Uso	Req (TUCP)	Req (h.h)	A&P (TUCP)	A&P (h.h)	Cod (TUCP)	Cod (h.h)	Tst (TUCP)	Tst (h.h)	TUCP TOTAL	TUCP (h.h)
UC_01	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_02	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_03	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_04	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_05	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_06	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_07	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_08	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_09	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_10	6,8	111,1	8,5	138,9	15,3	250,0	3,4	55,5	34,1	555,4
UC_11	5,7	92,6	7,1	115,7	12,8	208,3	2,8	46,3	28,4	462,9
UC_12	6,8	111,1	8,5	138,9	15,3	250,0	3,4	55,5	34,1	555,4
UC_13	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_14	5,7	92,6	7,1	115,7	12,8	208,3	2,8	46,3	28,4	462,9
UC_15	3,4	55,5	4,3	69,4	7,7	125,0	1,7	27,8	17,0	277,7
UC_16	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_17	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_18	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_19	5,7	92,6	7,1	115,7	12,8	208,3	2,8	46,3	28,4	462,9
UC_20	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_21	5,7	92,6	7,1	115,7	12,8	208,3	2,8	46,3	28,4	462,9
UC_22	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
UC_23	1,1	18,5	1,4	23,1	2,6	41,7	0,6	9,3	5,7	92,6
UC_24	2,3	37,0	2,8	46,3	5,1	83,3	1,1	18,5	11,4	185,1
Total	**70,43**	**1.147,93**	**88,03**	**1.434,91**	**158,46**	**2.582,84**	**35,21**	**573,96**	**352,13**	**5.739,64**

Além do tamanho do caso de uso, pode-se obter ainda uma granularidade mais fina, por meio do cálculo do tamanho de cada caso de uso por etapa do ciclo de vida (seção 5.5). A

partir da *Eq*. 5.7 e do *percentual de esforço* da Tabela 6.1, é calculado o tamanho do caso de uso UC_01 para as seguintes etapas do ciclo de vida:

- Requisitos: $TUCP_{(UC\_01\_Req)} = 5,7 * 0,20 \cong 1,1$
- Análise & Projeto: $TUCP_{(UC\_01\_A\ \&\ P)} = 5,7 * 0,25 \cong 1,4$
- Codificação: $TUCP_{(UC\_01\_Cod)} = 5,7 * 0,45 \cong 2,6$
- Teste: $TUCP_{(UC\_01\_Tst)} = 5,7 * 0,10 \cong 0,6$

A seguir, é calculado o tamanho de cada etapa do ciclo de vida do *Projeto A* através da *Eq*. 5.9:

- Requisitos: $TUCP_{(Req)} = 352,1 * 0,20 \cong 70,4$
- Análise & Projeto: $TUCP_{(A\ \&\ P)} = 352,1 * 0,25 \cong 88,0$
- Codificação: $TUCP_{(Cod)} = 352,1 * 0,45 \cong 158,5$
- Teste: $TUCP_{(Tst)} = 352,1 * 0,10 \cong 35,2$

Cálculo do Esforço

A partir do tamanho do caso de uso por etapa de ciclo de vida, pode ser obtido o esforço desse caso de uso por etapa de ciclo de vida (*Eq*. 5.12). A produtividade utilizada no *Projeto A* foi de 20 h.h, pois a equipe era experiente e já tinha conhecimento sobre o negócio do sistema. O esforço por etapa do ciclo de vida para o caso de uso UC_01 é calculado abaixo (em h.h):

- Requisitos: $Esforço_{(UC\_01\_Req)} = 1,1 * 0,815 * 20 \cong 18,5$
- Análise & Projeto: $Esforço_{(UC\_01\_A\ \&\ P)} = 1,4 * 0,815 * 20 \cong 23,1$
- Codificação: $Esforço_{(UC\_01\_Cod)} = 2,6 * 0,815 * 20 \cong 41,7$
- Teste: $Esforço_{(UC\_01\_Tst)} = 0,6 * 0,815 * 20 \cong 9,3$

O esforço do caso de uso UC_01 pode ser calculado de acordo com a *Eq*. 5.12, como se segue (em h.h):

$$Esforço_{(UC\_01)} = TUCP_{(UC\_01)} * EF * PROD = 5,7 * 0,815 * 20 \cong 92,6$$

O esforço pode ser calculado por etapa do ciclo de vida pela *Eq*. 5.12 (em h.h). Neste caso, também foi considerada a produtividade de cada etapa como sendo igual a 20h.h.

- Requisitos: $\text{Esforço}_{(\text{Req})} = 70{,}43 * 0{,}815 * 20 \cong 1.147{,}93$
- Análise & Projeto: $\text{Esforço}_{(\text{A \& P})} = 88{,}03 * 0{,}815 * 20 \cong 1.434{,}91$
- Codificação: $\text{Esforço}_{(\text{Cod})} = 158{,}46 * 0{,}815 * 20 \cong 2582{,}84$
- Teste: $\text{Esforço}_{(\text{Tst})} = 35{,}21 * 0{,}815 * 20 \cong 573{,}96$

O esforço total do *Projeto A* é dado pela *Eq*. 5.11 (em h.h), como se segue:

$$\text{Esforço} = 352{,}13 * 0{,}815 * 20 \cong 5.739{,}64$$

A produtividade utilizada para o cálculo do esforço do projeto pode ser obtida a partir da média entre as produtividades das etapas do ciclo de vida.

A seguir, os valores de esforço estimado e real baseados na TUCP e as produtividades de cada etapa do ciclo de vida são apresentados na Tabela 6.6. O valor do percentual por esforço, na coluna % Esforço Real, foi calculado a partir do esforço real despendido no *Projeto A* (Vide seção 6.1). Os valores do percentual estimado, utilizado para calcular as estimativas em TUCP apresentados, foram obtidos do banco de dados de estimativas da organização.

Tabela 6.6 - Dados do *Projeto A* calculados com base na TUCP

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	70,4	1147,9	956,2	20	13,58	18,19	-20
Análise e Projeto	88,0	1434,9	1007,8	20	11,45	19,17	-42
Codificação	158,5	2582,8	2734,5	20	17,26	52,02	6
Testes	35,2	574,0	558,0	20	15,85	10,62	-3
Total	**352,1**	**5739,6**	**5256,5**	**20**	**14,93**	**100**	**-9**

Na *Etapa de Requisitos*, seus produtos de trabalho foram todos desenvolvidos, isto é, as especificações de requisitos funcionais, não funcionais, as especificações de casos de uso, etc. A produtividade calibrada para tal etapa foi de 20 h.h. Neste projeto, o *líder de requisitos* era bastante experiente e dominava o problema, o que pode ser percebido na Tabela 6.6, em que a produtividade foi muito mais alta do que a prevista. Isto foi favorecido também, porque o projeto apresentou vários casos de usos simples de serem especificados, como casos de uso do tipo consulta.

A *Etapa de Análise & Projeto* foi bastante simplificada, porque o cliente deste Projeto disponibilizou um componente através de uma API Java para a manipulação de dados no ambiente *Web*. Além disso, muitas das classes e vários diagramas de seqüências já estavam modelados.

Na *Etapa de Codificação*, a equipe de projeto teve dificuldades com o entendimento dos métodos apresentados pelo componente, além de verificar erros de implementação do próprio componente, resultando em um esforço muito maior do que o planejado. Além disso, desligaram-se dois programadores da equipe do projeto no início das atividades de codificação, que estavam alocados em tempo integral, possuíam conhecimento do negócio a ser desenvolvido, e tinham alto domínio na linguagem de programação do projeto. Eles foram substituídos por outros três programadores. Um deles foi alocado em tempo parcial, e os outros dois não tinham domínio do negócio nem possuíam bom conhecimento na linguagem de programação utilizada no projeto.

Baseada nos fatos ocorridos, uma reestimativa poderia ter sido realizada no início das atividades de codificação deste projeto, o que não foi feito. Esta nova estimativa poderia ter sido realizada, alterando os fatores ambientais (EF), já que as características do ambiente foram afetadas. A Tabela 6.7 mostra esta simulação: o valor de EF, que antes era de 19,5 (Tabela 6.3), passou a ser de 15,5.

Tabela 6.7 - Novos fatores de complexidade ambiental do *Projeto A*

Fator	Valor	Descrição	Peso	Ponderação
F1	1,5	Familiar com o Processo do Institucional	3	4,5
F2	0,5	Experiência com a aplicação	2	1,0
F3	1,0	Experiência em orientação a objetos	3	3,0
F4	0,5	Liderança Técnica	2	1,0
F5	1,0	Motivação	4	4,0
F6	2,0	Requisitos estáveis	2	4,0
F7	-1,0	Integrantes em tempo parcial	1	-1,0
F8	-1,0	Linguagem de programação complexa	1	-1,0
				15,5

Como EF influencia diretamente no cálculo do esforço, ter-se-ia uma nova estimativa para a atividade de codificação. Recalculando-se o esforço para a *etapa de codificação*, ter-se-ia um novo esforço de 2.963,1 h.h., e não 2.582,84h.h como havia sido planejado anteriormente.

Como o esforço real do *Projeto A* para a etapa de codificação foi de 2.734,5h.h., isso mostra que, se esta reestimativa tivesse sido feita, o cliente teria conhecimento prévio de um possível atraso na entrega do produto. Além disso, ações de contingência poderiam ter sido disparadas, para que os impactos do atraso tivessem sido mitigados. Até uma renegociação com o cliente sobre uma nova data de entrega do produto poderia ter sido realizada.

Uma outra forma de se reestimar a etapa de codificação poderia ter sido feita com o aumento do valor de produtividade (PROD) para essa etapa. A produtividade, que antes da

saída dos recursos era de 20 h.h, poderia ter passado para 22 h.h, por exemplo. Assim, aplicando-se esta nova produtividade, a estimativa de esforço para a etapa de codificação seria de 2.841,12 h.h, ficando muito próximo do real:

$$\text{Esforço}_{(\text{Cod})} = 158{,}46 * 0{,}815 * 22 \cong 2.841{,}12$$

Na *Etapa de Teste*, o *Projeto A* não realizou as atividades de testes unitários, por falta de planejamento. A etapa de testes foi um dos pontos mais difíceis deste Projeto. O cliente era muito exigente quando recebia os produtos do projeto. Diante de qualquer erro encontrado durante os testes de campo, o cliente não aceitava o produto como entregue. Além disso, a alocação de recursos para a realização dos testes sistêmicos era escassa e isso fez com que os testes se estendessem mais do que o previsto.

UCP x TUCP

O *Projeto A* foi calculado também na técnica UCP (Tabela 6.8), e seus dados foram comparados com a técnica TUCP (Tabela 6.9).

O valor do tamanho do *Projeto A* em UCP é calculado na seqüência abaixo de acordo com a Tabela 6.2, Tabela 6.3 e Tabela 6.4.

$$\text{UUCP} = \sum \text{UAW} + \sum \text{UUCW} = 3 + 240 = 243$$

$$\text{UCP} = \text{UUCP} * \text{TCF} * \text{EF} = 243 * 1{,}125 * 0{,}815 = 222{,}8$$

O esforço estimado em UCP é calculado abaixo, considerando-se o mesmo valor de produtividade de 20 h.h, pois a equipe do projeto era experiente.

$$\text{ESFORÇO} = \text{UCP} * \text{PROD} = 222{,}8 * 20 = 4.457$$

Tabela 6.8 - Dados do *Projeto A* em UCP

Tamanho (UCP)	Esforço Estimado UCP	Esforço Real	% Erro Estimado
222,8	**4457,0**	**5256,5**	**18**

Tabela 6.9 - Dados do *Projeto A* em TUCP

Tamanho (TUCP)	Esforço Estimado TUCP	Esforço Real	
352,1	**5739,6**	**5256,5**	**- 9**

A Tabela 6.8 mostra o esforço estimado pela UCP juntamente com o esforço real despendido. Neste caso, o esforço real foi 18% superior ao estimado pela UCP.

A Tabela 6.9 apresenta o esforço estimado pela técnica TUCP, com um erro estimado de 9% menor que o esforço real produzido. Assim sendo, neste projeto a TUCP foi mais precisa que a UCP.

6.3 Projeto B

O *Projeto B* desenvolveu um produto de software em uma interface Web na plataforma J2EE.

A equipe de projeto era composta de dez membros. O coordenador deste projeto foi alocado apenas para as atividades de planejamento e acompanhamento. Essa equipe era composta por 50% de pessoas recém chegadas na empresa que não tinham domínio do processo de desenvolvimento organizacional. Além disso, três dos recursos estavam alocados em tempo parcial no projeto.

Além do grupo de testes, havia neste projeto um recurso alocado exclusivamente para as atividades de testes internos. Esse recurso era responsável por executar os testes de integração, sistêmicos antes de serem enviados ao grupo de teste, além dos testes de carga, performance e desempenho. Os testes unitários foram desenvolvidos e executados pelos próprios desenvolvedores. Uma outra questão importante foi a automação de muitos testes, exigindo um significativo esforço com relação à aprendizagem e ao manuseio da ferramenta de automação de testes na geração de seus *scripts*.

Este projeto utilizou um *framework* no processo de desenvolvimento e sua equipe precisou ser treinada para sua utilização, havendo um esforço adicional para a aprendizagem do mesmo.

Fatores de Complexidade Técnica

Os Fatores de Complexidade Técnica (TCF) do *Projeto B* estão descritos na Tabela 6.10. A seguir, serão mostrados os dados do primeiro incremento do *Projeto B*.

Tabela 6.10 - Fatores de complexidade técnica do *Projeto B*

Fator	Peso	Descrição	Valor	Ponderação
T1	2,0	Sistema distribuído	5	10,0
T2	1,0	Questões de desempenho de resposta (*throughputi*)	4	4,0
T3	1,0	Eficiência de usuário final (*online*)	4	4,0
T4	1,0	Processamento interno complexo	4	4,0
T5	1,0	Código deve ser reutilizável	5	5,0
T6	0,5	Fácil de instalar	3	1,5
T7	0,5	Fácil de usar	4	2,0
T8	2,0	Portável	5	10,0
T9	1,0	Fácil de modificar	5	5,0
T10	1,0	Concorrente	5	5,0
T11	1,0	Características especiais de segurança	4	4,0
T12	1,0	Acesso direto a componentes de terceiros	5	5,0
T13	1,0	Facilidades especiais de treinamento de usuários requeridas	3	3,0
TFator				**62,5**

O cálculo do TCF do *Projeto B* é calculado abaixo:

$$TCF = 0,6 + (0,01 * 62,5) = 1,225$$

Fatores de Complexidade Ambiental

Os Fatores de Complexidade Ambiental (EF) relacionados ao *Projeto B* estão descritos na Tabela 6.11.

Tabela 6.11 - Fatores de complexidade ambiental do *Projeto B*

Fator	Peso	Descrição	Valor	Ponderação
F1	1,5	Familiar com o Processo do Institucional	3	4,5
F2	0,5	Experiência com a aplicação	2	1,0
F3	1,0	Experiência em orientação a objetos	5	5,0
F4	0,5	Liderança Técnica	4	2,0
F5	1,0	Motivação	5	5,0
F6	2,0	Requisitos estáveis	2	4,0
F7	-1,0	Integrantes em tempo parcial	3	-3,0
F8	-1,0	Linguagem de programação complexa	2	-2,0
EFactor				**16,5**

O fator de complexidade ambiental (EF) para o *Projeto B* é dado por:

$$EF = 1,4 + (-0,03 * 16,5) \cong 0,905$$

Cálculo da TUCP

O *Projeto B* desenvolveu seis casos de usos para o primeiro incremento, onde três eram simples, dois eram intermediários e um era *n*-complexo (Tabela 6.12).

Tabela 6.12 - Ponderação de casos de uso por transações

Caso de Uso	Transações	Nº Transações	Peso (UUCW)	Peso (TUUCW)	TUCP
UC_01	simples	2T	5	5	6,8
UC_02	simples	3T	5	10	6,8
UC_03	intermediário	5T	10	5	13,6
UC_04	intermediário	5T	10	5	13,6
UC_05	***n*-complexo**	**12T**	**15**	**20**	**27,2**
UC_06	simples	2T	5	10	6,8
TOTAL		**29T**	**50**	**55**	**75**

O valor calculado para o somatório dos pesos dos atores foi igual a $\sum UAW = 6.$

Como $TUUCP = \sum UAW + \sum TUUCW$, então, $TUUCP = 6 + 55 = 61.$

Portanto, a $TUCP = 61 * 1,125 \cong 75.$

Na Tabela 6.13, serão apresentados os valores dos tamanhos em TUCP por caso de uso e por etapa do ciclo de vida, além do esforço (h.h) para cada caso de uso, e por etapa de ciclo de vida.

Tabela 6.13 - Distribuição de tamanho (TUCP) e esforço (h.h) por etapa do ciclo de vida

Caso de Uso	REQ (TUCP)	REQ (h.h)	A&P (TUCP)	A&P (h.h)	COD (TUCP)	COD (h.h)	TST (TUCP)	TST (h.h)	TUCP TOTAL	TUCP (h.h)
UC_01	1,4	24,6	1,7	30,7	3,1	55,3	0,7	12,3	6,8	123,0
UC_02	1,4	24,6	1,7	30,7	3,1	55,3	0,7	12,3	6,8	123,0
UC_03	2,7	49,2	3,4	61,5	6,1	110,7	1,4	24,6	13,6	245,9
UC_04	2,7	49,2	3,4	61,5	6,1	110,7	1,4	24,6	13,6	245,9
UC_05	5,4	98,4	6,8	123,0	12,2	221,3	2,7	49,2	27,2	491,8
UC_06	1,4	24,6	1,7	30,7	3,1	55,3	0,7	12,3	6,8	123,0
Total	**14,95**	**270,50**	**18,68**	**338,13**	**33,63**	**608,64**	**7,47**	**135,25**	**74,73**	**1352,52**

Cálculo do Esforço

Na Tabela 6.14, são apresentados os valores de esforço estimado e esforço real em TUCP e as produtividades para cada etapa do ciclo de vida considerada.

Tabela 6.14 - Dados do *Projeto B* calculados com base na TUCP

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	14,9	270,5	285,0	20	19,07	19,30	5
Análise e Projeto	18,7	338,1	196,0	20	10,49	13,27	-73
Codificação	33,6	608,6	852,0	20	25,34	57,68	40
Testes	7,5	135,3	144,0	20	19,27	9,75	6
Total	**74,7**	**1352,5**	**1477,0**	**20**	**19,77**	**100**	**9**

A produtividade estimada de 20 h.h para todas as etapas do ciclo de vida foi estabelecida pelo coordenador do projeto, pois os recursos previstos inicialmente para a alocação eram altamente especializados para executar tais atividades.

Alguns pontos foram observados ao término deste incremento. Na *Etapa de Requisitos*, foi gasto mais tempo para o levantamento dos requisitos do que o planejado. Isto foi ocasionado devido a constantes negociações para a aprovação final das especificações de requisitos.

Na *Etapa de Análise & Projeto*, não houve a necessidade de geração de muitos de seus artefatos; apenas os diagramas de classes foram efetivamente elaborados. Por este motivo, o esforço real foi muito menor do que havia sido estimado.

A *Etapa de Codificação* apresentou um esforço real maior que o estimado, que se deveu a dois motivos: (*i*) implementação dos testes unitários realizados pelos desenvolvedores; e (*ii*) esforço com relação à aprendizagem na utilização do *framework* utilizado no projeto.

Já para a *Etapa de Teste* obteve-se um esforço real um pouco maior do que o estimado. A introdução de uma ferramenta de automatização dos testes implicou em um esforço inicial, que posteriormente foi compensado.

Na Tabela 6.15, os dados do *Projeto B* foram recalculados para contemplar os valores das produtividades de cada etapa do ciclo de vida, refletindo-se mais as características do projeto. No entanto, esta calibração deve ser usada de maneira criteriosa, pois, ao se alterar a produtividade de cada etapa, a estimativa final do projeto também será alterada.

Tabela 6.15 - Dados do *Projeto B* recalculados com base na produtividade

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	14,9	324,6	285,0	24	19,07	19,30	-12
Análise e Projeto	18,7	253,6	196,0	15	10,49	13,27	-29
Codificação	33,6	852,1	852,0	28	25,34	57,68	0
Testes	7,5	162,3	144,0	24	19,27	9,75	-13
Total	**74,7**	**1592,6**	**1477,0**	**22,75**	**19,77**	**100**	**-7**

Na Tabela 6.16, os dados do *Projeto B* foram recalculados com base no *percentual de esforço*. Os valores utilizados não foram baseados nos *percentuais de esforço* da organização (Tabela 6.1), mas nas características do projeto, conhecidas pelo coordenador do projeto. Neste contexto, o *percentual de esforço* de cada etapa foi: Req = 20%; A&P = 15%; Cód = 55%; e Tst = 10%.

Tabela 6.16 - Dados do *Projeto B* recalculados com base no percentual de esforço

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	14,9	270,5	285,0	20	19,07	19,30	5
Análise e Projeto	11,2	202,9	196,0	20	17,49	13,27	-4
Codificação	41,1	743,9	852,0	20	20,73	57,68	15
Testes	7,5	135,3	144,0	20	19,27	9,75	6
Total	**74,7**	**1352,5**	**1477,0**	**20**	**19,77**	**100**	**9**

UCP x TUCP

O *Projeto B* foi calculado também na técnica UCP (Tabela 6.17), e seus dados são comparados com a técnica TUCP (Tabela 6.18).

O valor do tamanho deste projeto em UCP é calculado na seqüência abaixo segundo a Tabela 6.10, Tabela 6.11 e Tabela 6.12.

$$UUCP = \sum UAW + \sum UUCW = 6 + 50 = 56$$

$$UCP = UUCP * TCF * EF = 56 * 1,125 * 0,905 = 62,08$$

O esforço estimado em UCP é calculado abaixo, considerando-se o mesmo valor de produtividade de 20 h.h, pois a equipe do projeto era experiente.

$$ESFORÇO = UCP * PROD = 62,08 * 20 = 1.241,7$$

Tabela 6.17 - Dados do *Projeto B* em UCP

Tamanho (UCP)	Esforço Estimado UCP	Esforço Real	% Erro Estimado
62,08	1.241,7	1.477,0	19

Tabela 6.18 - Dados do *Projeto B* em TUCP

Tamanho (TUCP)	Esforço Estimado TUCP	Esforço Real	% Erro Estimado
74,7	1.352,5	1.477,0	9

A Tabela 6.17 mostra o esforço estimado pela UCP juntamente com o esforço real. Neste caso, o esforço real foi 19% superior se estimado pela UCP. A Tabela 6.18 apresenta o esforço estimado pela TUCP, que foi 9% superior ao esforço real. Assim sendo, neste projeto, a TUCP foi mais precisa que a UCP.

6.4 Projeto C

O *Projeto C* também desenvolveu um produto de software em uma interface Web na plataforma J2EE. Neste estudo de caso, será levado em consideração apenas o primeiro incremento, com o desenvolvimento de oito casos de uso.

Foi prevista a alocação de dez recursos humanos para o projeto, incluindo os papéis de coordenador de projeto e dos líderes de requisitos e configuração. O coordenador do projeto executava as atividades de planejamento e acompanhamento do projeto. O líder de configuração gastou 50% de seu tempo com as atividades de gerência de configuração. O líder de requisitos foi alocado totalmente nas atividades de gestão de requisitos.

O *Projeto C* passou por sérios problemas de alocação de recursos, pois durante todo o primeiro incremento houve uma alta rotatividade (*turning*) de membros da equipe (entradas e saídas constantes), o que influiu diretamente na produtividade da equipe.

Utilizou também um *framework* para dar suporte e aumentar a produtividade no desenvolvimento do sistema. No entanto, a equipe foi treinada durante este incremento para a utilização de referido *framework*. Com isto, a produtividade foi influenciada pela falta de experiência devido à curva de aprendizagem. Assim, para este incremento, a utilização do *framework* não resultou em aumento de produtividade da equipe do projeto.

Outro ponto crítico referiu-se à utilização de recursos de IHC (Interface Humano-Computador), o que influenciou o esforço da análise e projeto, pois o *web designer* era pouco experiente e houve demora na criação do *layout* das telas do sistema.

A seguir serão mostrados os dados do primeiro incremento referentes ao *Projeto C*.

Fatores de Complexidade Técnica

Os Fatores de Complexidade Técnica (TCF) relacionados ao *Projeto C* estão descritos na Tabela 6.19.

Tabela 6.19 - Fatores de complexidade técnica do *Projeto C*

Fator	Peso	Descrição	Valor	Ponderação
T1	2,0	Sistema distribuído	2	4,0
T2	1,0	Questões de desempenho de resposta (*throughputi*)	5	5,0
T3	1,0	Eficiência de usuário final (*online*)	5	5,0
T4	1,0	Processamento interno complexo	2	2,0
T5	1,0	Código deve ser reutilizável	2	2,0
T6	0,5	Fácil de instalar	2	1,0
T7	0,5	Fácil de usar	5	2,5
T8	2,0	Portável	2	4,0
T9	1,0	Fácil de modificar	2	2,0
T10	1,0	Concorrente	3	3,0
T11	1,0	Características especiais de segurança	3	3,0
T12	1,0	Acesso direto a componentes de terceiros	3	3,0
T13	1,0	Facilidades especiais de treinamento de usuários requeridas	0	0,0
TFator				**36,5**

O cálculo do TCF do *Projeto C* é calculado abaixo:

$$TCF = 0,6 + (0,01 * 36,5) = 0,965$$

Fatores de Complexidade Ambiental

Os Fatores de Complexidade Ambiental (EF) relacionados ao *Projeto C* estão descritos na Tabela 6.20.

Tabela 6.20 - Fatores de complexidade ambiental do *Projeto C*

Fator	Peso	Descrição	Valor	Ponderação
F1	1,5	Familiar com o Processo do Institucional	5	7,5
F2	0,5	Experiência com a aplicação	2	1,0
F3	1,0	Experiência em orientação a objetos	5	5,0
F4	0,5	Liderança Técnica	3	1,5
F5	1,0	Motivação	3	3,0
F6	2,0	Requisitos estáveis	4	8,0
F7	-1,0	Integrantes em tempo parcial	0	0,0
F8	-1,0	Linguagem de programação complexa	4	-4,0
EFactor				**22**

O fator de complexidade ambiental (EF) para o Projeto C é dado por:

$$EF = 1,4 + (-0,03 \text{ x } 22) \cong 0,74$$

Cálculo da TUCP

O *Projeto C* desenvolveu oito casos de usos no primeiro incremento, onde um era simples, três eram intermediários e quatro eram complexos (Tabela 6.21).

Tabela 6.21 - Ponderação de casos de uso por transações

Caso de Uso	*Transações*	*Nº Transações*	*Peso* (UUCW)	*Peso* (TUUCW)	*TUCP*
UC_01	simples	3T	5	5	5
UC_02	complexo	9T	15	15	16
UC_03	intermediário	7T	10	10	11
UC_04	complexo	8T	15	15	16
UC_05	intermediário	7T	10	10	11
UC_06	complexo	9T	15	15	16
UC_07	intermediário	7T	10	10	11
UC_08	complexo	11T	15	15	16
Total		**61T**	**95**	**95**	**103,3**

O valor calculado para o somatório dos pesos dos atores foi igual a $\sum UAW = 12$

Como $TUUCP = \sum UAW + \sum TUUCW$, então $TUUCP = 12 + 95 = 107$.

Portanto, a $TUCP = 107 * 0{,}965 \cong 103{,}3$.

Na Tabela 6.22, serão apresentados os valores dos tamanhos em TUCP por caso de uso e por etapa do ciclo de vida, além do esforço (h.h) para cada caso de uso, e por etapa de ciclo de vida.

Tabela 6.22 - Distribuição de tamanho (TUCP) e esforço (h.h) por etapa do ciclo de vida

Caso de Uso	REQ (TUCP)	REQ (h.h)	A&P (TUCP)	A&P (h.h)	COD (TUCP)	COD (h.h)	TST (TUCP)	TST (h.h)	TUCP TOTAL	TUCP (h.h)
UC_01	1,1	19,3	1,4	24,1	2,4	43,4	0,5	9,7	5,4	96,5
UC_02	3,3	57,9	4,1	72,4	7,3	130,3	1,6	29,0	16,3	289,5
UC_03	2,2	38,6	2,7	48,3	4,9	86,9	1,1	19,3	10,9	193,0
UC_04	3,3	57,9	4,1	72,4	7,3	130,3	1,6	29,0	16,3	289,5
UC_05	2,2	38,6	2,7	48,3	4,9	86,9	1,1	19,3	10,9	193,0
UC_06	3,3	57,9	4,1	72,4	7,3	130,3	1,6	29,0	16,3	289,5
UC_07	2,2	38,6	2,7	48,3	4,9	86,9	1,1	19,3	10,9	193,0
UC_08	3,3	57,9	4,1	72,4	7,3	130,3	1,6	29,0	16,3	289,5
Total	**20,65**	**366,76**	**25,81**	**458,45**	**46,46**	**825,21**	**10,33**	**183,38**	**103,26**	**1.833,81**

Cálculo do Esforço

Na Tabela 6.23, são apresentados os valores do esforço estimado e do esforço real em TUCP e as produtividades para cada etapa do ciclo de vida considerada.

Tabela 6.23 - Dados do *Projeto C* calculados com base na TUCP

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	20,7	366,8	212,81	24	10,31	11,37	-72
Análise e Projeto	25,8	458,5	547,25	24	21,20	29,25	19
Codificação	46,5	825,2	985,2	24	21,20	52,66	19
Testes	10,3	183,4	125,78	24	12,18	6,72	-46
Total	**103,26**	**1833,81**	**1871,0**	**24**	**18,12**	**100**	**2**

A produtividade estimada de 24 h.h para todas as etapas do ciclo de vida foi estabelecida pelo coordenador de *Projeto C*, pois era sabido que a equipe, apesar de experiente, iria utilizar um *framework* novo durante todo o ciclo de vida do projeto; assim, a produtividade do projeto seria menor. Com a finalização do primeiro incremento desse projeto, algumas justificativas esclarecem os valores encontrados na tabela acima.

A *Etapa de Requisitos* apresentou um desvio alto em relação ao estimado, devido às características dos casos de uso deste primeiro incremento, que eram muito semelhantes entre si. Portanto, o líder de requisitos reutilizou muito das especificações dos casos de uso e a produtividade foi alta.

Na *Etapa de Análise & Projeto*, a produtividade da equipe foi baixa devido ao uso do *framework*, pois a equipe não tinha experiência nesta ferramenta e a curva de aprendizagem influenciou a produtividade do projeto. Isto se refletiu também em retrabalho da equipe na construção dos diagramas de classes e do modelo de dados, pois a maneira como os diagramas estavam sendo modelados não estava de acordo com os padrões do *framework* utilizado. Outra questão que também influenciou esta etapa foi a criação da arte visual e a elaboração de interfaces (IHC). A construção das telas pelo *web designer* levou mais tempo do que o esperado.

Na *Etapa de Codificação*, a produtividade da equipe também foi baixa devido à utilização do *framework*, justificada pela inexperiência da equipe na utilização do mesmo, uma alta rotatividade (*turning*) de membros da equipe nesta etapa, além do desenvolvimento de testes unitários em algumas classes pré-estabelecidas. Além disso, as correções de defeitos influenciaram também no atraso da codificação.

Na *Etapa de Teste*, a produtividade real foi maior do que a estimada, devido às características dos casos de uso, pois, como eram muito semelhantes, facilitou sobremaneira a realização dos testes.

Baseado nas características do *Projeto C* apresentadas acima, o coordenador do projeto poderia ter estimado uma produtividade alta para as etapas de requisitos e de testes, conforme a Tabela 6.24.

Tabela 6.24 - Dados do *Projeto C* recalculados com base na produtividade estimada

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	20,7	305,6	212,81	20	10,31	11,37	-44
Análise e Projeto	25,8	534,9	547,25	28	21,20	29,25	2
Codificação	46,5	962,7	985,2	28	21,20	52,66	2
Testes	10,3	152,8	125,78	20	12,18	6,72	-21
Total	**103,26**	**1956,06**	**1871,0**	**24**	**18,12**	**100**	**-4**

UCP x TUCP

O Projeto C foi calculado também na técnica UCP (Tabela 6.25) e seus dados são comparados com a técnica TUCP (Tabela 6.26).

O valor do tamanho do Projeto C em UCP é calculado na seqüência abaixo de acordo com a Tabela 6.17, Tabela 6.18 e Tabela 6.19.

$$UUCP = \sum UAW + \sum UUCW = 12 + 95 = 107$$

$$UCP = UUCP * TCF * EF = 107 * 0{,}965 * 0{,}74 = 76{,}41$$

O esforço estimado em UCP é calculado abaixo, considerando-se o mesmo valor de produtividade de 20 h.h, pois a equipe do projeto era experiente.

$$ESFORÇO = UCP * PROD = 76{,}41 * 24 = 1.833{,}81$$

Tabela 6.25 - Dados do *Projeto C* em UCP

Tamanho (UCP)	Esforço Estimado UCP	Esforço Real	% Erro Estimado
76,41	**1.833,81**	**1.871,0**	**2**

Tabela 6.26 - Dados do *Projeto C* em TUCP

Tamanho (TUCP)	Esforço Estimado TUCP	Esforço Real	% Erro Estimado
103,26	**1.833,81**	**1.871,0**	**2**

A Tabela 6.25 mostra o esforço estimado pela UCP juntamente com o esforço real. Neste caso, o esforço real foi 2% superior ao estimado pela UCP. A Tabela 6.26 apresenta o esforço estimado pela TUCP, que também foi 2% superior ao esforço real. Assim sendo, neste projeto, a TUCP teve a mesma precisão do que a UCP.

Note-se que, neste incremento do *Projeto C*, não houve nenhum caso de uso do tipo *n*-complexo. Portanto, o esforço para desenvolver o incremento foi o mesmo tanto para UCP como para TUCP.

Assim, pode-se concluir que, para projetos onde não haja casos de uso *n*-complexos, o esforço estimado é o mesmo entre as duas técnicas, quando se utiliza a mesma produtividade para o projeto como um todo (todas as etapas têm a mesma produtividade).

Isto vem a comprovar que a TUCP não alterou a técnica UCP em sua essência, mas propôs uma extensão para tratar melhor os casos de uso muito complexos, para que não fossem subestimados, além de melhorar as estimativas por etapa do ciclo de vida.

6.4.1 Acompanhamento do Progresso Funcional

O que se observa na prática é que casos de uso, freqüentemente, apresentam complexidade diferente e necessitam de níveis de esforços diferentes para a sua realização.

O objetivo do acompanhamento do progresso funcional (seção 5.10) do primeiro incremento do *Projeto C* é mostrar o aumento percentual da funcionalidade desse incremento, em diferentes períodos de seu desenvolvimento.

A avaliação de progresso do primeiro incremento do *Projeto C* focalizou os aspectos funcionais incorporados ao sistema, ou seja, o acompanhamento de $\mu_{Sistema}$. A Figura 6.2 apresenta a variação do progresso funcional do *Projeto C* durante o período em que seu desenvolvimento foi monitorado.

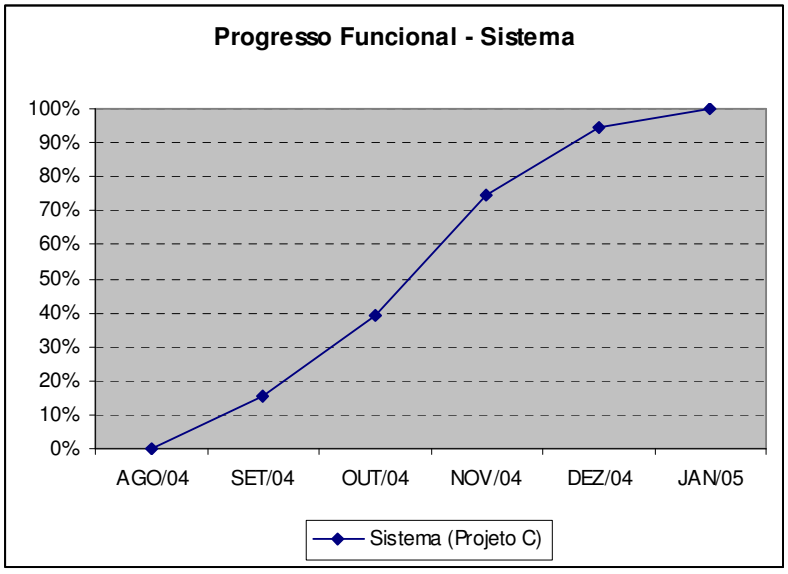


Figura 6.2 – Progresso funcional do primeiro incremento do *Projeto C*

Os dados relativos ao progresso funcional durante todo o primeiro incremento do *Projeto C* foram obtidos através dos dados reportados nas reuniões de acompanhamento pelos membros da equipe do projeto e da recuperação de informações registradas do projeto. Além

do progresso do primeiro incremento como um todo, foi monitorado também o progresso de cada etapa do ciclo de vida e de cada caso de uso relacionado a esse incremento.

A Figura 6.3 apresenta os resultados referentes ao progresso funcional de cada etapa do ciclo de vida do primeiro incremento do *Projeto C*.

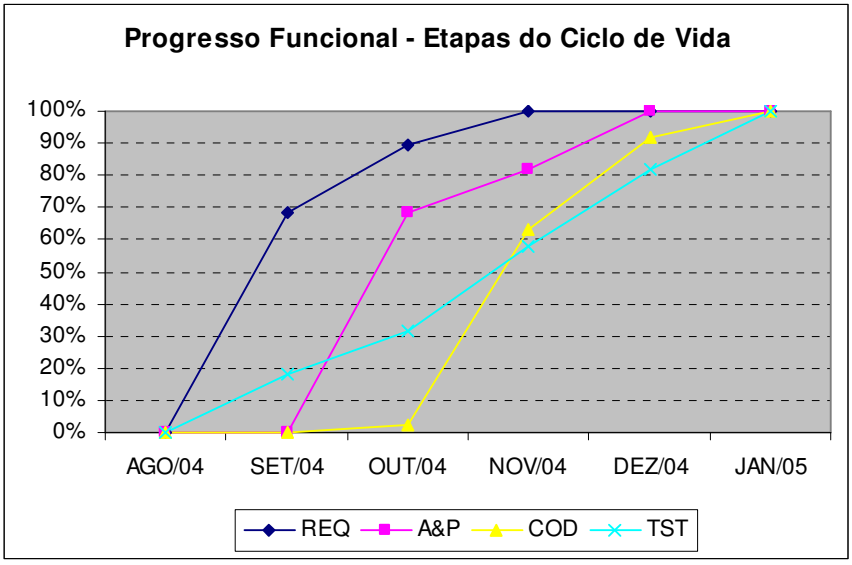


Figura 6.3 - Progresso das etapas do ciclo de vida do primeiro incremento do *Projeto C*

A Figura 6.4 exibe os resultados obtidos no acompanhamento de três casos de uso (UC_01, UC_03, e UC_06) deste incremento, considerados como casos de uso simples intermediário e complexo, respectivamente.

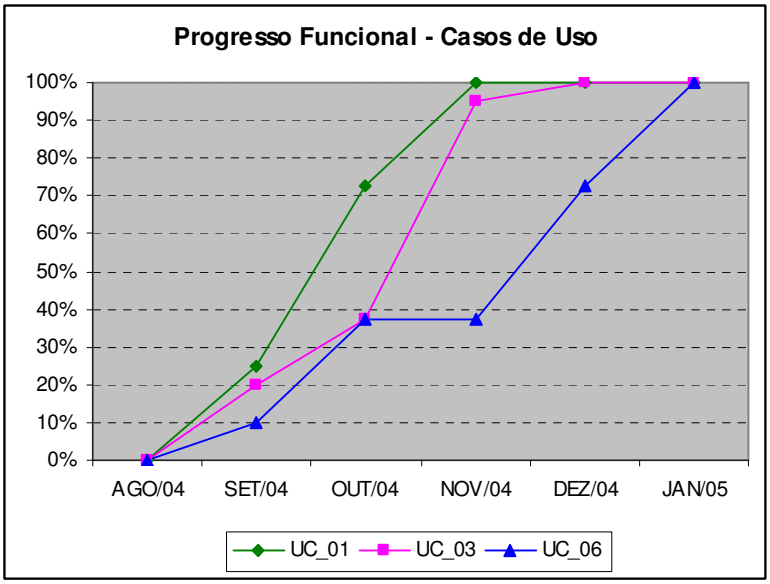


Figura 6.4 - Progresso dos casos de uso do primeiro incremento do *Projeto C*

As seguir, serão apresentados os resultados obtidos durante as cinco avaliações de progresso funcional realizadas entre os meses de agosto de 2004 e janeiro de 2005, correspondendo ao período do primeiro incremento do *Projeto C*. Essas avaliações ocorreram em intervalos de aproximadamente um mês.

Primeira avaliação do progresso funcional

O primeiro incremento do *Projeto C* foi iniciado em agosto de 2004. Após um mês da data de início do projeto, foi realizado o acompanhamento do progresso funcional deste incremento, conforme Tabela 6.27. Pode ser observado que as atividades de teste já vinham sendo realizadas após a finalização de algumas especificações de casos de uso. Pode também ser notado que mais da metade do trabalho referente à etapa de requisitos havia sido realizada até este período.

Tabela 6.27 - Progresso funcional obtido na primeira avaliação

Caso de Uso	REQ	μUC_Etapa	A&P	μUC_Etapa	COD	μUC_Etapa	TST	μUC_Etapa	μUC
UC_01	1,1	1,0	1,4	0,0	2,4	0,0	0,5	0,5	25
UC_02	3,3	0,5	4,1	0,0	7,3	0,0	1,6	0,0	10
UC_03	2,2	1,0	2,7	0,0	4,9	0,0	1,1	0,0	20
UC_04	3,3	1,0	4,1	0,0	7,3	0,0	1,6	0,5	25
UC_05	2,2	0,5	2,7	0,0	4,9	0,0	1,1	0,0	10
UC_06	3,3	0,5	4,1	0,0	7,3	0,0	1,6	0,0	10
UC_07	2,2	0,0	2,7	0,0	4,9	0,0	1,1	0,0	0
UC_08	3,3	1,0	4,1	0,0	7,3	0,0	1,6	0,5	25

Etapa	μEtapa
REQ	68,4
A&P	0,0
COD	0,0
TST	18,4

μSistema
15,53

Segunda avaliação do progresso funcional

Em um segundo momento da análise do progresso funcional deste incremento (Tabela 6.28) foi verificado que as atividades de análise e projeto (A&P) já tinham sido iniciadas e mais da metade dessas atividades já haviam sido realizadas. Apenas um dos casos de uso não tinha sido iniciado ainda sua etapa de requisitos, devido à indefinição de alguns dados a serem fornecidos pelo cliente. Todos os outros casos de uso restantes já haviam concluído a etapa de requisitos. Nesta análise, parte da implementação foi iniciada.

Tabela 6.28 - Progresso funcional obtido na segunda avaliação

Caso de Uso	REQ	μUC_Etapa	A&P	μUC_Etapa	COD	μUC_Etapa	TST	μUC_Etapa	μUC
UC_01	1,1	1,0	1,4	1,0	2,4	0,5	0,5	0,5	72,5
UC_02	3,3	1,0	4,1	0,5	7,3	0,0	1,6	0,0	32,5
UC_03	2,2	1,0	2,7	0,5	4,9	0,0	1,1	0,5	37,5
UC_04	3,3	1,0	4,1	1,0	7,3	0,0	1,6	0,5	50
UC_05	2,2	1,0	2,7	1,0	4,9	0,0	1,1	0,0	45
UC_06	3,3	1,0	4,1	0,5	7,3	0,0	1,6	0,5	37,5
UC_07	2,2	0,0	2,7	0,0	4,9	0,0	1,1	0,0	0
UC_08	3,3	1,0	4,1	1,0	7,3	0,0	1,6	0,5	50

Etapa	μEtapa
REQ	89,5
A&P	68,4
COD	2,6
TST	31,6

μSistema
39,34

Terceira avaliação do progresso funcional

Na Tabela 6.29, pode ser observado que as atividades da etapa de requisitos foram totalmente concluídas. As atividades de análise & projeto, codificação, e teste melhoraram seu progresso funcional, especialmente na etapa de codificação em relação à avaliação anterior, saindo de 2,6% para 63%. O progresso funcional deste incremento já atingiu quase 75%.

Tabela 6.29 - Progresso funcional obtido na terceira avaliação

Caso de Uso	REQ	μUC_Etapa	A&P	μUC_Etapa	COD	μUC_Etapa	TST	μUC_Etapa	μUC
UC_01	1,1	1,0	1,4	1,0	2,4	1,0	0,5	1,0	100
UC_02	3,3	1,0	4,1	1,0	7,3	1,0	1,6	1,0	100
UC_03	2,2	1,0	2,7	1,0	4,9	1,0	1,1	0,5	95
UC_04	3,3	1,0	4,1	1,0	7,3	1,0	1,6	0,5	95
UC_05	2,2	1,0	2,7	1,0	4,9	0,0	1,1	0,0	45
UC_06	3,3	1,0	4,1	0,5	7,3	0,0	1,6	0,5	37,5
UC_07	2,2	1,0	2,7	0,0	4,9	0,0	1,1	0,0	20
UC_08	3,3	1,0	4,1	1,0	7,3	1,0	1,6	1,0	100

Etapa	μEtapa
REQ	100,0
A&P	81,6
COD	63,2
TST	57,9

μSistema
74,61

Quarta avaliação do progresso funcional

A quarta avaliação do progresso funcional foi realizada em dezembro de 2004 e pode ser observada na Tabela 6.30 que o progresso do sistema passou de 74,6% para 94,6%, depois de aproximadamente um mês de projeto. Houve atrasos na codificação de alguns casos de uso, mas este desvio não foi tão significativo.

Tabela 6.30 - Progresso funcional obtido na quarta avaliação

Caso de Uso	REQ	μUC_Etapa	A&P	μUC_Etapa	COD	μUC_Etapa	TST	μUC_Etapa	μUC
UC_01	1,1	1,0	1,4	1,0	2,4	1,0	0,5	1,0	100
UC_02	3,3	1,0	4,1	1,0	7,3	1,0	1,6	1,0	100
UC_03	2,2	1,0	2,7	1,0	4,9	1,0	1,1	1,0	100
UC_04	3,3	1,0	4,1	1,0	7,3	1,0	1,6	1,0	100
UC_05	2,2	1,0	2,7	1,0	4,9	1,0	1,1	0,5	95
UC_06	3,3	1,0	4,1	1,0	7,3	0,5	1,6	0,5	72,5
UC_07	2,2	1,0	2,7	1,0	4,9	1,0	1,1	0,5	95
UC_08	3,3	1,0	4,1	1,0	7,3	1,0	1,6	1,0	100

Etapa	μEtapa
REQ	100,0
A&P	100,0
COD	92,1
TST	81,6

μSistema
94,61

Quinta avaliação do progresso funcional

Em janeiro de 2005, o primeiro incremento do *Projeto C* foi entregue, tendo todas suas etapas concluídas com sucesso e um progresso funcional de 100% para os casos de uso, as etapas e o sistema.

Através destas avaliações, obteve-se uma melhor visão da sensibilidade da métrica de progresso funcional do projeto, levando-se em consideração o progresso em termos do esforço necessário para finalizar o projeto, observando o que foi feito e o que ainda falta realizar.

Todavia, esta maneira de avaliação do progresso funcional depende, e muito, da participação efetiva e da colaboração de todos os envolvidos no projeto, especialmente os analistas e desenvolvedores, que detêm conhecimento dos artefatos produzidos ao longo do ciclo vida, e podem fornecer as informações mais acuradas sobre o progresso dos mesmos.

6.5 Projeto D

O Projeto D teve como foco o desenvolvimento de um sistema de software *web*, com a linguagem J2EE, utilizando a tecnologia EJB (*Enterprise Java Beans*), conforme os seguintes padrões de codificação: *Xdoclet*, *Junit*, *Cactus*, *Mock Objects*. Neste projeto foram utilizados tanto componentes (comerciais) adquiridos, quanto componentes fornecidos pelo cliente.

Este projeto foi divido em quatro iterações do ciclo de vida iterativo-incremental. O *primeiro incremento* consistiria de uma pré-concepção do sistema, isto é, da definição dos requisitos e do modelo de casos de uso para todo o sistema, e da definição de uma arquitetura inicial, que serviria de base para a prova de conceitos, antecipando assim possíveis riscos que poderiam surgir no incremento seguinte.

O *segundo e terceiro incrementos* consistiriam na estabilização da arquitetura, baseada na prova de conceitos realizada no incremento anterior e no refinamento da arquitetura, além do detalhamento dos casos de uso do sistema, e a sua implementação. Nesses dois incrementos os testes seriam realizados sob demanda.

O *quarto incremento* consistiria na homologação do sistema. O foco desta etapa era a realização de testes e ciclos de correção de erros indicados pelo cliente.

Este projeto foi de grande porte com, em média, 25 pessoas alocadas, tendo chegado até 36 pessoas durante uma determinada etapa do ciclo de desenvolvimento. Estavam alocados para este projeto inicialmente: um gerente de projeto, um coordenador, um líder de requisitos, um líder de configuração, dois analistas de negócios, dois arquitetos, quatro projetistas, um *web designer*, um documentador, um analista de banco de dados, treze

analistas de sistema e dois analistas de testes. Nem todos os recursos foram alocados com dedicação exclusiva para o projeto.

Diversos problemas surgiram no decorrer do processo de desenvolvimento de software deste projeto. Como conseqüências, ao final do segundo incremento, o projeto foi abortado. Os motivos que levaram ao fracasso deste projeto não foram considerados nas estimativas e nas propostas iniciais com o cliente.

A seguir, são elencados alguns pontos ao longo do processo de desenvolvimento, que culminaram no insucesso do *Projeto D*:

- *Gerência de Projeto*:
  - ✓ Erros nas estimativas iniciais do projeto
  - ✓ Alocação de equipe inexperiente
  - ✓ Alocações parciais
  - ✓ Tempo limitado para treinamentos e aprendizados
  - ✓ Reestimativas prejudicadas
  - ✓ Replanejamentos contínuos sem bases sólidas
  - ✓ Resolução dos problemas com riscos assumidos unilateralmente
  - ✓ Falta de informações impediu maior clareza e foco nas negociações com o cliente.
- *Requisitos*:
  - ✓ Diferentes níveis de detalhamento dos casos de uso
  - ✓ Dinâmica ineficaz na validação dos casos de uso
  - ✓ Aumento da complexidade dos casos de uso
- *Análise e Projeto*:
  - ✓ Retrabalho na construção de protótipos decorrente do atraso no fechamento dos casos de uso
  - ✓ Baixa qualidade na entrega das prova de conceito, prejudicando a definição da arquitetura do sistema
  - ✓ Diferenças de entendimento entre os grupos de arquitetura
  - ✓ Desgastes na identificação e resolução de problemas
- *Codificação*:
  - ✓ Falta de aderência aos guias do projeto
  - ✓ Documentação resumida dos componentes do cliente
  - ✓ Equipe inexperiente
  - ✓ Testes unitários insuficientes

- ✓ Critério subjetivo de conclusão da atividade
- ✓ Retrabalho na implementação dos casos de uso
- ✓ Baixa produtividade
- ✓ Baixa qualidade do código entregue.

- *Testes*:
  - ✓ Equipe inexperiente
  - ✓ Diferenças de entendimento entre os grupos de analistas de testes;
  - ✓ Retrabalho na elaboração dos casos de testes
  - ✓ Alta complexidade nos casos de testes
  - ✓ Baixa produtividade da equipe
  - ✓ Baixa qualidade dos testes realizados
  - ✓ Uso de novas ferramentas de testes (testadores pouco experientes nessas ferramentas).

Além das questões citadas acima, houve dificuldades da equipe em seguir padrões e processos estabelecidos, além da falta de maturidade da equipe nas novas tecnologias utilizadas no projeto. O treinamento nessas tecnologias foi realizado de forma rápida e não houve tempo hábil para sua assimilação.

O *Projeto D* foi inicialmente estimado utilizando-se apenas a técnica UCP, num total de 750,5 UCPs, conforme a Tabela 6.31. Nesse projeto, deveriam ser desenvolvidos 45 casos de uso, sendo que 22 seriam implementados no segundo incremento, e os casos de uso restantes no terceiro incremento.

Entretanto, no decorrer do primeiro incremento foi constatado que os casos de uso não eram tão simples de ser desenvolvidos como havia sido previsto na etapa inicial. O detalhamento do negócio era muito mais complexo do que o imaginado. Foi através deste projeto que se constatou efetivamente que os casos de uso do tipo *complexo* poderiam ter sido subestimados, pois existiam casos de uso com um número muito alto de transações, o que deveria ter sido considerado.

Tabela 6.31 - Dados do *Projeto D* calculados com base na UCP

Projeto D	Tamanho (UCP)	Esforço Estimado (h.h)
Total	**750,5**	**13.628,7**

Verificando-se que o *Projeto D* havia sido subestimado, pois já apresentava atrasos consideráveis, uma nova reestimativa foi realizada levando-se em consideração os casos de uso *n*-complexos, propostos pela TUCP, segundo a Tabela 6.32.

Tabela 6.32 - Dados do *Projeto D* reestimados com base na TUCP

Etapas	Tamanho (TUCP)	Esforço Estimado (H.H)
Requisitos	243,3	3893,1
Análise e Projeto	304,1	4866,3
Codificação	547,5	11387,2
Testes	121,7	1946,5
Total	**1.216,6**	**22.093,1**

A produtividade utilizada nas estimativas acima foi alta, pois as pessoas inicialmente alocadas para a execução deste projeto eram especialistas para cada tipo de função. No entanto, a preocupação maior era com a etapa de codificação, que utilizaria novas tecnologias. Assim, a produtividade utilizada foi de 20 h.h para requisitos, análise & projeto, e testes, e uma produtividade de 26 h.h para codificação. Entretanto, esta situação não se configurou de fato, devido aos problemas ocorridos citados anteriormente.

Os dados apresentados na Tabela 6.33 são referentes ao primeiro e segundo incrementos pertencentes ao *Projeto D*. Ao final do segundo incremento, esse projeto foi abortado.

Tabela 6.33 - Dados do 1º e 2º incrementos *Projeto D* com alta produtividade

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	123,0	1967,3	1835,0	20	14,92	10,00	-7
Análise e Projeto	153,7	2459,1	4540,2	20	29,54	24,75	85
Codificação	276,7	5754,3	9285,1	26	33,56	50,62	61
Testes	61,5	983,6	2682,7	20	43,64	14,63	173
Total	**614,8**	**11164,4**	**18343,0**	**21,5**	**29,84**	**100**	**64**

Na Tabela 6.33, a produtividade aplicada foi baseada nos valores iniciais da proposta, já que a equipe alocada era considerada produtiva. Todavia, isto não aconteceu na realidade, conforme comparadas na Tabela 6.33 as colunas "Produtividade Estimada (H.H/TUCP)" e "Produtividade Real (H.H/TUCP)".

Assim, para que o desvio do esforço estimado em relação ao esforço real fosse atenuado, a produtividade deveria ter sido revista, já que a equipe não era tão produtiva quanto o imaginado. Isto poderia ter diminuído o percentual de erro, conforme apresentado na Tabela 6.34.

Tabela 6.34 - Dados do 1º e 2º incrementos *Projeto D* com baixa produtividade

Etapa	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	123,0	2754,2	1835,0	28	14,92	10,00	-33
Análise e Projeto	153,7	3442,8	4540,2	28	29,54	24,75	32
Codificação	276,7	6639,6	9285,1	30	33,56	50,62	40
Testes	61,5	1377,1	2682,7	28	43,64	14,63	95
Total	**614,8**	**14213,7**	**18343,0**	**28,5**	**29,84**	**100**	**29**

Algumas diferenças apresentadas nos valores dos esforços estimados e reais deste projeto já foram explicadas anteriormente, atestando seu insucesso. Entretanto, outras características, como a alocação de *analistas de requisitos*, que eram especialistas nas regras de negócio do projeto, foram de suma importância na etapa de requisitos.

Na Tabela 6.35 e Tabela 6.36, são apresentados dados referentes ao primeiro e segundo incremento do *Projeto D*, calculados em UPC e TUCP, respectivamente, tendo sido utilizada uma produtividade de 28,5 h.h, valor próximo à média real.

Tabela 6.35 -Dados do 1º e 2º incrementos *Projeto D* calculado em UCP

Tamanho (UCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	% Erro Estimado
321,9	**9.303,0**	**18.343,0**	**97**

Tabela 6.36 - Dados do 1º e 2º incrementos *Projeto D* calculado em TUCP

Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	% Erro Estimado
614,8	**14.213,7**	**18.343,0**	**29**

Como pode ser observado nas duas tabelas acima, a TUCP, apesar de apresentar um percentual de erro alto em relação ao valor estimado (29%), ainda assim se mostrou mais acurada do que a técnica UCP, que tem um percentual de erro estimado de 97%.

O insucesso maior das estimativas do *Projeto D* foi devido, especialmente, a subestimativas realizadas para casos de uso de alta complexidade, como também ao valor inadequado atribuído à produtividade da equipe alocada para o projeto.

6.6 Projeto E

O *Projeto E* teve como foco o desenvolvimento de um sistema de software web na plataforma J2EE, utilizando a tecnologia EJB (*Enterprise Java Beans*), JSP, *Taglibs*. Foram utilizados os seguintes *frameworks* de codificação: *Junit*, *ibates*, *strutes*, *javamail*.

Este projeto consta de dois incrementos, mas apenas o primeiro incremento será objeto de estudo, já que o projeto ainda está em andamento, até o momento deste estudo de caso.

A equipe do projeto era composta de oito recursos humanos. Durante o primeiro incremento houve duas substituições de recursos. Essas substituições realizadas foram críticas durante o incremento. Como conseqüência, alguns casos de uso foram mal implementados, gerando vários erros e conseqüente retrabalho.

O entendimento dos *requisitos* deste projeto foi satisfatório já que o escopo definido pelo cliente estava claro e o negócio em si não era complexo. A *etapa de análise & projeto* foi rápida devido ao claro entendimento dos requisitos. No entanto, ao final do incremento, diagnosticou-se que essa etapa havia sido simplificada demasiadamente, gerando grandes dificuldades na *etapa de codificação*.

A etapa de codificação foi demorada, pois além dos casos de uso do sistema a serem desenvolvidos, foram implementados dois componentes para dar suporte à aplicação. Foi também utilizada parte de um *enginneing* de busca adquirido de terceiros e adaptado à realidade do projeto.

O *enginneing* (o "elemento de reúso") não era conhecido por nenhum integrante da equipe; com isso, houve uma curva de aprendizagem considerável, até que suas funções fossem assimiladas e pudessem ajudar no desenvolvimento do projeto.

Diversos erros surgiram durante a etapa de codificação, resultando em retrabalho no suporte ao código do *enginneing*. Apesar disso, o relato dos desenvolvedores foi de que o "elemento de reúso" ajudou bastante ao longo do projeto, pois se o mesmo fosse desenvolvido dentro da aplicação, levaria muito tempo, já que sua lógica era muito complexa.

A entrada de novos recursos no projeto demandou um esforço maior durante a etapa de codificação, já que esses recursos não conheciam os *frameworks* utilizados no projeto, levando um certo tempo de aprendizagem.

Uma dificuldade adicional foi encontrada nos casos de uso complexos. Como a análise & projeto desses casos de uso havia sido simplificada, sua implementação tornou-se difícil, e muitos erros foram inevitáveis, gerando retrabalho.

Além disso, a etapa de codificação do sistema demandou grande retrabalho devido a erros encontrados na etapa de testes. Durante essa etapa, apesar dos requisitos terem sido de fácil entendimento, a lógica da implementação era complexa.

Vários erros também surgiram na etapa de testes devido à não execução de testes de integração e unitários com a freqüência adequada para o porte do projeto. Grande parte desta atividade ficou acumulada para o final do incremento, gerando muitos erros.

Foram desenvolvidos onze casos de uso para este primeiro incremento. O grau de dificuldade para construir os componentes e adaptar o *enginneing* não foi refletido diretamente nas estimativas realizadas, pois foi assumido que eles estariam acoplados ao sistema, o que já estaria refletindo os valores estimados dos casos de uso do projeto.

A seguir, serão apresentados os cálculos das estimativas para o primeiro incremento do *Projeto E*.

Fatores de Complexidade Técnica

Os valores obtidos para os fatores de complexidade técnicos (TCF) relacionados ao *Projeto E* estão apresentados na Tabela 6.37.

Tabela 6.37 - Fatores de complexidade técnica do *Projeto E*

Fator	Peso	Descrição	Valor	Ponderação
T1	2,0	Sistema distribuído	3	6,0
T2	1,0	Questões de desempenho de resposta (*throughputi*)	3	3,0
T3	1,0	Eficiência de usuário final (*online*)	4	4,0
T4	1,0	Processamento interno complexo	3	3,0
T5	1,0	Código deve ser reutilizável	4	4,0
T6	0,5	Fácil de instalar	5	2,5
T7	0,5	Fácil de usar	5	2,5
T8	2,0	Portável	4	8,0
T9	1,0	Fácil de modificar	4	4,0
T10	1,0	Concorrente	1	1,0
T11	1,0	Características especiais de segurança	3	3,0
T12	1,0	Acesso direto a componentes de terceiros	2	2,0
T13	1,0	Facilidades especiais de treinamento de usuários requeridas	4	4,0
TFator				**47**

O cálculo do TCF do Projeto E é calculado abaixo:

$$TCF = 0,6 + (0,01 * 47) = 1,07$$

Fatores de Complexidade Ambiental

Os valores obtidos para os fatores de complexidade ambiental (EF) relacionado ao Projeto E estão descritos na Tabela 6.38.

Tabela 6.38 - Fatores de Complexidade Ambiental do *Projeto E*

Fator	Peso	Descrição	Valor	Ponderação
F1	1,5	Familiar com o Processo do Institucional	3	4,5
F2	0,5	Experiência com a aplicação	3	1,5
F3	1,0	Experiência em orientação a objetos	3	3,0
F4	0,5	Liderança Técnica	3	1,5
F5	1,0	Motivação	2	2,0
F6	2,0	Requisitos estáveis	3	6,0
F7	-1,0	Integrantes em tempo parcial	2	-2,0
F8	-1,0	Linguagem de programação complexa	3	-3,0
EFactor				**13,5**

O fator de complexidade ambiental (EF) para o *Projeto E* é dado por:

$$EF = 1,4 + (-0,03 \text{ x } 13,5) \cong 0,99 .$$

Cálculo da TUCP

Para o primeiro incremento, o *Projeto E* desenvolveu 11 casos de usos, sendo dois simples, três intermediários, cinco complexos, e um *n*-complexo (Tabela 6.39).

Tabela 6.39 - Tabela de ponderação de casos de uso por transações

Caso de Uso	Transações	Nº Transações	Peso (UUCW)	Peso (TUUCW)	TUCP
UC_01	complexo	10 T	15	15	17
UC_02	simples	3T	5	5	6
UC_03	simples	3T	5	5	6
UC_04	complexo	8T	15	15	17
UC_05	complexo	10T	15	15	17
UC_06	***n*-complexo**	**13T**	**15**	**20**	**22**
UC_07	complexo	10T	15	15	17
UC_08	intermediário	7T	10	10	11
UC_09	intermediário	7T	10	10	11
UC_10	complexo	8T	15	15	17
UC_11	intermediário	4T	10	10	11
TOTAL		**83T**	**130**	**135**	**151**

O valor calculado para o somatório dos pesos dos atores foi igual a $\sum \text{UAW} = 6.$

Como $\text{TUUCP} = \sum \text{UAW} + \sum \text{TUUCW}$, então, $\text{TUUCP} = 6 + 135 = 141.$

Portanto, a $\text{TUCP} = 141 * 1,07 \cong 150,87$.

Na Tabela 6.40, serão apresentados os valores dos tamanhos em TUCP por caso de uso e por etapa do ciclo de vida, além do esforço (h.h) para cada caso de uso, e por etapa de ciclo de vida.

Tabela 6.40 - Distribuição de tamanho (TUCP) e esforço (h.h) por etapa do ciclo de vida

Caso de Uso	REQ (TUCP)	REQ (h.h)	A&P (TUCP)	A&P (h.h)	COD (TUCP)	COD (h.h)	TST (TUCP)	TST (h.h)	TUCP TOTAL	TUCP (h.h)
UC_01	3,4	56,7	4,2	83,4	7,5	210,2	1,7	28,4	16,8	378,6
UC_02	1,1	18,9	1,4	27,8	2,5	70,1	0,6	9,5	5,6	126,2
UC_03	1,1	18,9	1,4	27,8	2,5	70,1	0,6	9,5	5,6	126,2
UC_04	3,4	56,7	4,2	83,4	7,5	210,2	1,7	28,4	16,8	378,6
UC_05	3,4	56,7	4,2	83,4	7,5	210,2	1,7	28,4	16,8	378,6
UC_06	4,5	75,6	5,6	111,2	10,1	280,2	2,2	37,8	22,4	504,8
UC_07	3,4	56,7	4,2	83,4	7,5	210,2	1,7	28,4	16,8	378,6
UC_08	2,2	37,8	2,8	55,6	5,0	140,1	1,1	18,9	11,2	252,4
UC_09	2,2	37,8	2,8	55,6	5,0	140,1	1,1	18,9	11,2	252,4
UC_10	3,4	56,7	4,2	83,4	7,5	210,2	1,7	28,4	16,8	378,6
UC_11	2,2	37,8	2,8	55,6	5,0	140,1	1,1	18,9	11,2	252,4
Total	**30,17**	**510,39**	**37,72**	**750,58**	**67,89**	**1.891,46**	**15,09**	**255,20**	**150,87**	**3.407,63**

Cálculo do Esforço

Na Tabela 6.41, são apresentados os valores de esforço estimado e esforço real em TUCP e as produtividades para cada etapa do ciclo de vida considerada.

Tabela 6.41 - Dados do *Projeto E* calculados com base na TUCP

Etapas	Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	Produtividade Estimada (H.H/TUCP)	Produtividade Real (H.H/TUCP)	% Esforço Real	% Erro Estimado
Requisitos	30,2	510,4	606,5	17	20,10	16,51	19
Análise e Projeto	37,7	750,6	475,7	20	12,61	12,95	-58
Codificação	67,9	1891,5	2289,2	28	33,72	62,32	21
Testes	15,1	255,2	302,0	17	20,02	8,22	18
Total	**150,9**	**3407,6**	**3673,4**	**20,5**	**21,61**	**100**	**8**

Conforme mostrado na Tabela 6.41, os dados apresentados refletem bem a realidade deste incremento, onde os valores reais para a etapa de *análise & projeto* foi bem menor que o estimado, o que afetou as outras etapas de alguma forma. O esforço despendido nas etapas de *codificação* e *testes* foi maior do que o estimado.

Apesar de o *Projeto E* ter utilizado "elementos de reúso" (componentes, *frameworks*, etc.), que auxiliaram na *etapa de codificação*, o esforço real foi maior que o estimado devido principalmente a dois componentes novos que foram desenvolvidos, e um componente de *engineeing* que foi adaptado para o projeto. O valor estimado de cada um deles já estava inicialmente refletido nos próprios casos de uso do projeto. Ao final do primeiro incremento, foram analisados os resultados e chegou-se à conclusão que os dois componentes a serem desenvolvidos e o *engineeing* a ser adaptado para o projeto deveriam ter sido estimados isoladamente das funcionalidades do sistema.

Uma outra questão que justifica o desvio esforço real em relação ao estimado da *etapa de codificação* é a alta complexidade lógica dos casos de uso. A *análise & projeto* desses

casos de uso foi realizada de maneira muito simplificada (vide Tabela 6.41), resultando em dificuldades para os desenvolvedores durante a etapa de codificação deste incremento, caracterizada por dificuldades no entendimento dos casos de uso, e posterior retrabalho. Outro fator que contribuiu para este desvio foi o *turning* dos recursos no projeto.

Como conseqüência direta dos desvios da etapa de codificação, a *etapa de teste* também teve um desvio considerável do valor estimado, devido a muitos erros encontrados no código desenvolvido. Uma outra questão foi a deficiência na condução dos testes de integração e unitários. Foram deixados para o final do incremento todos os testes, o que sobrecarregou a etapa de teste.

A produtividade de cada etapa foi configurada de acordo com cada etapa do ciclo de vida, levando-se em consideração as potencialidades dos recursos alocados em cada etapa (coluna "Produtividade Estimada" da Tabela 6.41).

Na Tabela 6.42 e Tabela 6.43 são apresentados os dados do *Projeto E* calculados na técnica UCP e na TUCP. Como pode ser observado nas duas tabelas acima, a TUCP apresenta um percentual de erro em relação ao valor estimado de 8%, mostrando-se ainda mais precisa do que a técnica UCP, que teve um percentual de erro estimado de 24%.

Tabela 6.42 - Dados do *Projeto E* em UCP

Tamanho (UCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	% Erro Estimado
144,8	2968,2	3673,4	24

Tabela 6.43 - Dados do *Projeto E* em TUCP

Tamanho (TUCP)	Esforço Estimado (H.H)	Esforço Real (H.H)	% Erro Estimado
150,9	3407,6	3673,4	8

6.7 Comparação entre os Projetos Analisados

Neste capítulo, foram apresentados os resultados das estimativas realizadas em cinco projetos de desenvolvimento de software utilizando a técnica TUCP proposta.

Conforme a análise dos resultados dos experimentos realizados (Tabela 6.44), a utilização da técnica TUCP acarretou uma redução na margem de erro entre o esforço previsto e o esforço realizado em quatro dos cinco projetos em relação à UCP.

Apenas no Projeto C, a margem de erro entre as duas técnicas foi exatamente a mesma, isto é, 2%, o que se justifica por não haver nesse projeto nenhum caso de uso do tipo *n*-complexo.

Tabela 6.44 - Resumo das estimativas UCP e TUCP dos projetos estudados

Projeto	Tamanho UCP	Tamanho TUCP	Esforço Estimado UCP (H.H)	Esforço Estimado TUCP (H.H)	Esforço Real (H.H)	% Erro Estimado UCP	% Erro Estimado TUCP
A	222,8	352,1	4.457,0	5.739,6	5.256,5	18	-9
B	62,1	74,7	1.241,7	1.352,5	1.477,0	19	9
C	76,4	103,3	1.833,8	1.833,8	1.871,0	2	2
D	321,9	614,8	9.303,0	14.213,7	18.343,0	97	29
E	144,8	150,9	2.968,2	3.407,6	3.673,4	24	8

Além disso, a utilização da TUCP leva a um melhor refinamento das estimativas ao longo das etapas do ciclo de vida do projeto, uma vez que é usual se ter grupos de pessoas com diferentes graus de produtividade em cada uma dessas etapas.

O acompanhamento do progresso funcional ao longo das etapas do ciclo de vida utilizado pela TUPC pode auxiliar especialmente o coordenador do projeto no gerenciamento de suas atividades no decorrer do longo do processo de desenvolvimento.

6.8 Conclusão

Este capítulo apresentou a aplicação da técnica TUCP, que é uma extensão da UCP, em cinco projetos de software em uma organização certificada CMM, nível 2, que está se preparando para ser certificada CMMI, nível 3. A TUCP mostrou-se mais acurada que a UCP nos resultados obtidos.

O próximo capítulo apresenta a conclusão deste trabalho, suas contribuições e trabalhos futuros.

Capítulo **7**

Conclusão

Este capítulo apresenta as principais conclusões deste trabalho, cujo objetivo foi propor uma extensão da UCP, a TUCP. Estão descritas suas contribuições e sugestões para trabalhos futuros.

Um dos aspectos mais positivos da técnica de Pontos por Casos de Uso (UCP) é a sua simplicidade, a sua facilidade de uso e a rapidez de ser aplicada, quando se têm as informações necessárias para realizar as estimativas (DAMODARAN & WASHINGTON, s.d). Essa técnica ainda é pouco utilizada na indústria, apesar de outras experiências de uso de UCP já ter sido realizada na indústria por Anda *et al*. (2001), por Ribu (2001) e por empresas como a *Rational*, SUN e IBM (PROBASCO, 2002).

Este trabalho apresentou a técnica TUCP, que é uma extensão da UCP. Nos cinco projetos do estudo de caso realizado em uma organização certificada CMM, nível 2, a TUCP apresentou um cálculo mais acurado para o esforço de projetos. Além disso, a TUCP permitiu uma visão mais detalhada de estimativas nas principais etapas do ciclo de vida do software, possibilitando um acompanhamento mais efetivo do projeto.

Podem-se destacar como principais contribuições deste trabalho, a TUCP, que é uma extensão da técnica UCP:

- Proposição de um *Guia para a elaboração de casos de uso*, uma vez que influencia fortemente no cálculo do tamanho dos casos de uso.
- Detalhamento do conceito de transação no contexto de casos de uso, por afetar diretamente o cálculo dos pontos de casos de uso não ajustados.
- Refinamento da contagem de pontos por casos de uso complexos, para evitar possíveis subestimativas, especialmente quando o número de transações é muito grande.

- Desatrelar os fatores técnicos de ambiente (EF) do cálculo do tamanho, por se entender que o tamanho é uma grandeza física que não deveria ter seu valor alterado em função dos fatores técnicos de ambiente.
- Considerar apenas no cálculo do Esforço os EFs, juntamente com o fator de produtividade (de forma semelhante à UCP).
- Granularizar o cálculo do tamanho do projeto por cada etapa do processo de desenvolvimento e por caso de uso. O somatório de todos esses tamanhos será o tamanho final de todo o projeto.
- Utilizar um fator de produtividade por etapa do ciclo de vida, gerando um percentual de erro menor em relação ao valor real.
- Tratar o progresso funcional do projeto através de suas etapas do ciclo de vida ou do próprio caso de uso.

Como conclusões mais importantes deste trabalho, apreendidas ao longo dos experimentos realizados, pode-se destacar que:

- Uma base histórica de estimativas da organização deve ser implementada (como recomenda a literatura de métricas de software), para que sirva de fundamentação nas calibrações e nos fatores de produtividade para projetos futuros.
- A inexistência de padrões aceitos universalmente para a especificação de casos de uso dificulta a comparação entre projetos de diferentes organizações. Portanto, não há como garantir que os valores em UCP ou TUCP estarão medindo a mesma coisa, se os critérios utilizados para a construção de casos de uso forem muito diversificados.
- A especificação de caso de uso deve ser descrita em um nível de detalhamento adequado, para que a estimativa baseada em UCP / TUCP seja eficaz.
- Os fatores de ambiente (EF) passaram a estar relacionados apenas com o esforço; assim, o tamanho passa a depender somente de requisitos funcionais e não-funcionais.

7.1 Trabalhos Futuros

Dentre os trabalhos futuros que podem dar continuidade à pesquisa aqui apresentada, sugerem-se:

- Proposta de modelagem de um sistema de métricas que suporte a técnica proposta neste trabalho, sendo aderente a modelos e normas como, por exemplo, ISO 15504, ISO 12207, e CMMI-SW.
- Elaboração de uma base histórica que dê suporte às estimativas de uma organização utilizando a TUCP, afim de que sirva de fundamentação nas calibrações e nos fatores de produtividade para projetos futuros a serem desenvolvidos.
- Consideração do reuso nas estimativas de tamanho das diversas etapas do ciclo de vida do projeto de software (requisitos, análise e projeto, implementação e testes), para projetos de software que utilizem padrões (*patterns*), componentes e *frameworks*.
- Proposta de definição para o progresso funcional em relação aos produtos de trabalho desenvolvidos por cada etapa do ciclo de vida, baseado no quão completos estão estes produtos de trabalho, não sendo mais apenas pela etapa, o que permite a granularidade tornar-se menor e mais precisa. Os produtos de trabalho (artefatos) corresponderiam aos documentos envolvidos com a produção do caso de uso e a incorporação das funcionalidades, por ele proposta, dentro do sistema.

7.2 Considerações Finais

O trabalho realizado buscou o refinamento das estimativas em projetos que utilizam a técnica UCP como base para o planejamento dos projetos de software, possibilitando através da TUCP um detalhamento das estimativas nas principais etapas do ciclo de vida do software com o objetivo de fornecer um gerenciamento do projeto mais efetivo e preciso, além de prover um acompanhamento do progresso de seus tamanhos.

Um processo que formalize a utilização de um conjunto de métricas interessa fortemente às grandes empresas, que necessitam de um maior controle sobre seus projetos e estão dispostas a investir para obter tais recursos.

REFERÊNCIA BIBLIOGRÁFICA

ABDEL-HAMID, T. **Software project control: an experimental investigation of judgment with fallible information**. IEEE Transacions on Software Engeneering. V. 19, n. 6, June, 1993.

ABRAN, A. et al. **Functional size measurement methods: COSMIC-FFP: design and field trials**. In: FESMA-AEMES Software Measurements Conference, 2000.

AGUIAR, M. **Pontos de função ou pontos de caso de uso? Como estimar projetos orientados a objetos**. Developer´s Magazine. V. 7, n. 77. Jan., 2003.

AHERN, D.; CLOUSE, A; TURNER, R. **CMMI Distilled: A Practical Introduction to Integrated Process Improvement**. SEI Series in Software Engineering, Addison-Wesley, 2001.

ALBRECHT, A.J. **Measuring Applications Development Productivity**. Proceedings of IBM Applic. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA. 1979.

ANDA, B; et al. **Estimating software development effort based on use cases: experiences from industry**. In: International Conference on the Unified Modeling Language (UML2001). Proceedings. Toronto, Oct. 1 – 5, 2001.

ANDA, B. **Comparing effort estimates based on use case points with expert estimates**. Empirical Assessment in Software Engineering (EASE 2002). Keele, UK, April, 8 – 10, 2002.

ANDRADE, E. **Pontos de casos de uso e pontos por função na gestão de estimativa de tamanho de projetos de software orientados a objetos**. Dissertação de Mestrado, Universidade Católica de Brasília. Fevereiro, 2004.

ARIFOGLU, A. **A methodology for software cost estimation**. Software Engineering Notes, vol. 18, 1993.

BASILI. V.; CALDIERA, G.; ROMBACH, H. **Goal Question Metric paradigm**. In:Encyclopedia of Software Engeneering. V. 2, 1994.

BOEHM, B. **Software engeneering economics**. Prentice Hall, 1981.

BOEHM, B. **Software cost estimation with COCOMO II**. Prentice Hall, New Jersey, 2000.

BONFIN, F. **Métricas de software**. Disponível em: <http://www.internext.com.br/mssa/medidas.html>.Acesso em:09/09/2004

BITTNER, K., SPENCE, I. **Use Case Modeling**. Addison Wesley, 2002

CHAMPEAUX D. **Object-Oriented Development Process and Metrics**. Prentice Hall, 1997.

CHRISSIS, M., et al. **CMMI: Guidelines for Process Integration and Product Improvement**. Addison-Wesley Pub Co, 2003.

COCKBURN, A. **Writing effective: use cases**. Addison-Wesley Boston, 2001.

CUNHA, G.; ALMEIDA, E. **Estimativa de projetos com base em casos de uso**. Trabalho apresentado ao Simpósio Internacional de Melhoria de Processos de Software (SIMPROS), Anais Recife, CenPRA/SENAC. Recife, 2003.

DAMODARAN, M; WASHINGTON A. **Estimation using use case points**. Computer Science Program. Texas – Victoria; University of Houston. s.d. Disponível em:<http://bfpug.com.br/Artigos/UCP/Damodaran-Estimation_Using_Use_Case_Points.pdf>. Acesso em: 11/04/2004

DEKKERS, C. **Pontos de função e medidas: o que é um ponto de função?**. Quality Plus Technologies, Inc., Publicado no QAI Journal. Dezembro/1998. Disponível em: <www.qualityplustech.com>. Acesso em: 11/06/2004.

DEKKERS, C. **Measuring the 'Logical' or 'Functional' size of software projects and software application**. ISO Bulletin, May, 2003.

DEMARCO, T. **Controle de projetos de software**. Editora Campus, 1989.

DEMARCO, T, "**The role of software development methodologies: past, present and future**". In: 18th Internacional Conference on Software Engineering, Berlim, Alemanha, Março, 1996.

DINSMORE I.,CAMPBELL P. **Como se tornar um profissional em gerenciamento de projetos**. Rio de Janeiro: Qualitymark. Ed., 2003.

DUMKE, R.; FOLTIN, E. **Applicability of full function point for Siemens AT**. Proceedings of the IWSM'99, Lc Superieur, Canada, 1999.

FEITOSA, C.; JANSEN, S. **Processo de estimativa e acompanhamento de tamanho e esforço para o ProSCes**. In: Encontro da Qualidade e produtividade em software (EQPS). Fortaleza, MCT, 2003.

FENTON, N.; NEIL, M. **Software Metrics and Risk**. FESMA, Amsterdam, 1999.

FENTON, N.; Neil, M. **Software Metrics: roadmap. Future of software engeneering.** Limerick, Ireland ACM, 2000.

FENTON, N.; NEIL, M.; PFLEEGER, S. **Software metrics: a rigorous & practical approach**. Chapman & Hall, 1991.

FENTON, N., NEIL, M.; PFLEEGER, S. **Software metrics: a rigorous & practical approach**. Boston: PWS Publishing Company, 1997.

FULL FUNCTION POINTS **(FFP). Full Function Points: Counting Practices Manual**. Technical Report, University of Quebec, Montreal, 1997.

FOWLER, M; Scott, K. **UML Essencial - Um breve guia para a linguagem-PAD**. Bookman, 2005.

FREIRE, H. **Calculando estimativas: o método de pontos de caso de uso**. S.d.

FUREY, C. **Why we should use function points**. IEEE. Mar/April, 1997.

GARMUS, D.; HERRON, D. **Function Point Analysis: measurement practices for successful software projects**. Addison-Wesley: EUA, 2000.

GLASS, R.L. **Facts and fallacies of software engineering**. Addison-Wesley, 2003

HARZAN, C. **Análise por ponto de função: uma abordagem gerencial**. Rio de Janeiro, SERPRO, 1999.

HELLER, R. **An introduction to function point analysis**. Journal of Defense Software Engineering, vol. 8, 1995.

HO, V. T.; ABRAN, A. **A framework for automating function point counting from source code**. Proceedings of The IWSM'99, Lac Superieur, Canada, 1999.

HUMPHREY, W.S. **Characterizing the software process: a maturity framework**. Software Engineering Institute, MU/SEI-87-TR- 11, ADA182895, June 1987.

HUMPHREY, W.S.; SWEET, W.L. **A method for assessing the software engineering capability of contractors**. Software Engineering Institute, CMU/SEI-87-TR-23, ADA187320. September 1987.

HUMPHREY, W. S., **Managing the Software Process**. Addison Wesley Longman. Inc.1991.

IEEE SOFTWARE ENGENEERING STANDARDS (IEEE). **IEEE Standard 610. 12-1990**. IEEE, 1993.

INTERNATIONAL FUNCTION POINT USERS GROUP (IFPUG). **Function Point Counting Practices Manual**. Version 4.1, January, 1999.

_____. **Function Point Counting Practices Manual**. Release 4.1. Ohio: IFPUG, 2000.

_____. **Function Point Counting Practices: Case Study 3 – Analysis**. Construction. Release 2.0. Princeton Junction: IFPUG, 2001.

INTERNATIONAL STANDARD ORGANIZATION (ISO). **ISO/IEC 9126**: Software Engineering – Product quality – Part 1: Quality model. 2001.

_____. **ISO/IEC 15504**: Information Technology: Process Assessment – Part 2.2003.

_____. **ISO/IEC 12207**: Tecnologia da Informação: processos de ciclo de vida de software. ABNT, Rio de Janeiro. 1998,

JACOBSEN; CHRISTERSON; OVERGAARD. **Object-oriented software engineering: a use case-driven approach**. Addison-Wesley, 1992.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The Unified Modeling Language user guide**. Addison-Wesley, 1998.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **UML: Guia do usuário**. Tradução Fábio Freitas da Silva. Rio de Janeiro: Campus, 2000.

JALOTE, P. **CMM in Practice: processes for executing software projects at infosys**. Addison-Wesley, 2001.

JONES, C. **Software challenges: function point: a new way of looking at tools**. Computer, August, 1994. Disponível em: <http://dlib2.computer.org/co/books/co1995/pdf/rx102.pdf>. Acesso em 24/02/03.

JONES, C. **What are function points?**. 1995.Disponível em: <http://home.iprimus.com.au/ian_cathy/Archive/CQU_files/85303/func_pts.html>. Acesso em 24/02/03.

JORGENSEN, M.; INDAHL, U. **Effort Estimation: software effort estimation by analogy and regression toward the mean**. 2003.

JURISON, J. **Software project management: the manager's view**. Communications of the Association for Information System V. 2, article 17, Sep. 1999.

KARNER, G. **Metrics for Objectory**. Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December, 1993.

KEMERER, C. F.; PORTER, B. S. **Improving the reability of function point measurement: an empirical study**. IEEE Transactions on Software Engineering, vol. 18, nº 11, 1992.

KRUCHTEN, P. **The Rational Unified Process: an introduction**. Addison-Wesley, 2001.

LAI, R.. "**The move to mature processes**". IEEE Software, pp. 14-17, July, 1993.

LARMAN, C. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos**. Trad. Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

LONGSTREET, B. **Use Case and Function Point**. Blue Springs: Longstreet Consulting Inc., 2000. Disponível em: <http://www.ifpug.com/fpafund.htm>. Acesso em 30/08/04.

McCONNELL, S. **Rapid Development – Taming Wild Software Schedules**. Microsoft Press, 1996.

McGARRY, J. et al. **Practical Software Mensurement: objective information for decision makers**. Boston: Addison-Wesley, 2001.

McPHEE, C. **SENG 621**: **Software process management: software size estimation**. University of Calgary. 1999. 11p. Disponível em: <http://sern.ucalgary.ca/~cmcphee/SENG621/Software_Size_Estimation.html>. Acesso em 18/08/04.

MENESES J. B.; MOURA H. P., **Um framework de avaliação de progresso de projetos de software orientado a objetos**. Workshop de Qualidade de Software (WQS2000), João Pessoa-PB, outubro 2000.

MENESES J. B.; MOURA H. P. **Inspector – Processo de avaliação de progresso de projetos de software orientado a objetos**. Disponível em http://www.cin.ufpe.br/~inspector, 2000. Acesso em 05/10/2004.

MENESES J. B.; MOURA H. P., **Inspector: Um Processo de avaliação de progresso para projetos de software**. Dissertação de Mestrado, Universidade Federal de Pernambuco (UFPE). Recife, 2001.

MISIC V.; TESIC D. **Downsizing the estimation of software quality: a small object-oriented case study**. In: Technology of Object-Oriented Languages and Systems. Proceedings S.d. Disponível em: <http://dlib2.computer.org/conferen/tools/9096/pdf/90960308.pdf>. Acesso em 24/10/2004.

MÖLLER, K.H.; PAULISH, D.J. **Software Metrics: a practioner's guide to improved product development**. IEEE Computer Society Press, Chapmam & Hall; 1993.

MONTEIRO, T.C.; PIRES, C.G.S.; BELCHIOR, A.D. **Estimations by Work Product Type: An extension of the UCP technique for the CMMI-SW level 2 and 3**. Metrics News issue, Berlin, Königs Wusterhausen, November 2-5, 2004. Disponível em: http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/mnews/news2004_1.pdf

MONTEIRO, T.C.; PIRES, C.G.S.; BELCHIOR, A.D. ***TUCP*: Uma Extensão da Técnica UCP**. IV Simpósio Brasileiro de Qualidade de Software – SBQS 2005, Porto Alegre - RS, Junho, 2005.

MONTEIRO, T.C.; PIRES, C.G.S.; BELCHIOR, A.D. **An extension of estimates based on UCP attending the necessities of SW-CMM level 2**. Proceedings of International Conference on Computing, Communications and Control Technologies - CCCT 2004, Austin (Texas), USA, August 14-17, 2004.

MONTEIRO, T.C.; PIRES, C.G.S.; BELCHIOR, A.D. **Estimativas por Tipo de Produto de Trabalho**. Proceedings of XXX Latin American Conference of Informatics – CLEI, Arequipa, Peru, September 27th, 2004 - October, 1st 2004.

MOREIRA, T.; RIOS, E. **Projeto e engenharia de software: teste de software**. Ed. Alta Book, 2003.

MYLIUS S.; MARODIN H. **O estabelecimento de estimativas em projetos**. S.d.

OBJECT MANAGEMENT GROUP, INC. (OMG). **Unified Modeling Language Specification**. Version 1.5, March, 2003. Disponível em: <http://www.omg.org>. Acesso em: 17/09/2004.

PÁDUA, F. W. **Engenharia de software: fundamentos, métodos e padrões**. 2 ed. Rio de Janeiro: LTC, 2003.

PAULK, M. C.; WEBER, C. V.; CURTIS, B.; CHRISSIS, M. B. **The Capability Maturity Model: guidelines for improving the software process**. Carnegie Mellon University, Software Engineering Institute, Addison-Wesley Longman Inc. 1997.

PAULK, M.C.; CURTIS, B.; CHRISSIS, M.B., et al. **Capability Maturity Model for Software**. Software Engineering Institute, CMU/SEI-91-TR- 24, ADA240603, Agust 1991.

PAULK, M.C. **Capability Maturity Model for Software**. Version 1.1, Technical Report CMU/SEI-93-TR-024, 1993.

PAULK, M.C. et. al. **Key Practices of the Capability Maturity Model**. Version 1.1, Technical Report CMU/SEI-93-TR-025, 1993.

PETERS, K. **Software project estimation. software productivity**. Vancouver, British Columbia, Canada.1999. Disponível em: <http://www.spd.ca/downloads/resources/estimates/estbasics.pdf>. Acesso em 12/11/2004.

PROJECT MANAGEMENT BODY OF KNOWLEGDE (PMBOK). Belo Horizonte: PMI-MG, (edição traduzida pelo PMI/MG). 2000.

POLLICE, G. **Measuring Up**. Copyright IBM Corporation 2004 – RationalEdge, August, 2004. Disponível em: <www-106.com/developerworks/rational/library/content/RationalEdge/aug04/5585.html>. Acesso em: 16/08/2004.

PRESSMAN R. S., **Software Engineering: A Practitioner's Approach**. McGraw-Hill, 5ª ed., 1999.

PRESSMAN, R. S. **Engenharia de Software**. 5.ed. Rio de Janeiro. McGraw-Hill, 2002.

PROBASCO, L. **Dear Dr. Use Case: what about Function Point and Use Cases?**. The Rational Edge 2002. Disponível em: <http://www.therationaledge.com/content/au_02/t_drUseCase_lp.jsp>. Acesso em 13/09/2004.

QUATRANI, T., **Visual modeling with Rational Rose 2000 and UML**. Addison-Wesley, 2ªed., outubro, 1999.

RAGLAND, B. **Measure, Metric or Indicator: What´s the Difference?**. Crosstalk, vol. 8, nº 3, March, 1995.

RATIONAL CORPORATION. **DEV112: Principles of Analysis I**. Web Courses. Disponível em: <http://www.rational.net>. Acesso em 23 de Abril de 2003.

RIBU, K. **Estimating Object-Oriented Software Projects with Use Cases**. Master of Sciense Thesis, University of Oslo, United States. November, 2001.

ROETZHEIM, W. H. **Estimating Internet Development**. Software Development Magazine, August, 2000.

ROSS, M. **Size does Matter: Continuous Size Estimating and Tracking. Quantitative Software Management**. Disponível em: <http://www.qsm.com>. Acesso em 18/08/04.

RULE, P.G. **Using measures to understand requirements**. Software Measurement Services Ltd, 2001.

BOOCH, G.; JACOBSON, I., RUMBAUGH, J. **Unified Modeling Language – User's Guide**. Addison-Wesley, 1999.

RATIONAL UNIFIED PROCESS®. Rational Software Corporation. Versão 2002.05.00. Copyright © 1987 – 2001.

_____. Rational Software Corporation. Copyright © 1987 – 2003.

SALVIANO, C. F.; et al. **Qualidade de software: Teoria e Prática**. Prentice Hall. 2001.

SANDERS, J., CURRAN, E. **Software Quality**. Dublin: ACM Press Books. 1994.

SCHNEIDER, G.; WINTERS, J. **Applying Use Case: A Practical Guide**. 2nd ed. Addison-Wesley, 2001.

SOFTWARE ENGENEERING INSTUTUTE (SEI). **CMMI-SW for Systems Engineering/Software Engineering**.Version 1.1 - CMU/SEI-2002-TR-012. 2002. Disponível em: <http://www.sei.cmu.edu/publications/documents/02.reports/02tr002.html>. Acessado em 27/04/2004.

SIMON, I. K. **Using function point analysis method or line of code for software size estimation?**. Proceedings of ESCOM-SCOPE 2000 Conference, Munique, Germany, 2000.

SMITH, L. **Function Point Analysis and Its Uses**.
Disponível em: <http://www.predicate.com/up_fp.html>. Acesso em 08/05/2004.

SOMMERVILLE, I.; KOTONIA, G. **Requirements Engineering Processes and Techniques**. Horizon Pubs & Distributors Inc, 1998.

SPARKS, S.; KASPCZYNSKI, K. **The art of sizing projects**. Sun World. S.d.

STANDISH GROUP. **The Chaos Report**. Disponível em:<http://www.standishgroup.com>. Acesso em 24/06/2005.

SYMONS, C.R. **Software Sizing and Estimating, MKII FPA**. John Wiley and Sons, 1991.

SYMONS, C.R. **Come back function point analysis (modernized) - all is forgiven!**. Software Measurement Services Ltd, 2001.

TAVARES, H. C. **Medição de Pontos por Função a partir de Especificação Orientada a Objetos**. Dissertação de Mestrado, Universidade Federal de Pernambuco (UFPE). Recife, 1999.

TSUKUMO, A. N et al. **Modelos de Processos de Software: Visão Global e Análise Comparativa**. Anais VII CITS - Conferência Internacional de Tecnologia de Software: Qualidade de Software, Curitiba,1996.

VALENÇA, A. **Estimativa de Esforço de Software Orientado a Objetos**. Recife. 2003.

VASQUEZ, C. E. **Análise de ponto de função: medição, estimativas e gerenciamento de projetos de software**. 1ed. São Paulo C.J.: Érica, 2003.

WEBER, C.V.; Paulk, M.C.; et al. **Key Practices of the Capabilities Maturity Model**. Software Engineering Institute (SEI), CMU/SEI-91-TR-25, ADA240604.August, 1991.

ZAHRAN, S., Software Process Improvement, Addison Wesley Longman Inc, 1998.

APÊNDICE A

Modelo de Especificação de Caso de Uso

<Nome do Projeto>

ESPECIFICAÇÃO DE CASO DE USO

(<ID> - <NOME DO CASO DE USO>)

VERSÃO <X.Y>

Responsável:	**E-mail:**
Elaborador(es):	**E-mail:**

ÍNDICE

1 HISTÓRICO

Data	Versão	Descrição	Responsável
<dd/mm/aaaa>	*<X.Y>*	*<detalhes>*	*<nome>*

2 INTRODUÇÃO

2.1 Objetivos

<Definir o propósito deste documento.>

2.2 Público alvo deste documento

- *<Lista do público alvo do documento>*

2.3 Glossário

<Listar o glossário relacionado a este documento.>

2.4 Referências

[1] <Ajustar conforme o documento>.

3 CASO DE USO

3.1 <Nome do caso de uso>

3.2 Descrição do caso de uso

<Descrição comportamental do caso de uso.>

3.3 Atores envolvidos

<Lista do nome e descrição dos atores associados ao caso de uso.>

3.3.1 Diagrama de hierarquia

<Inclusão do diagrama de hierarquia dos atores, quando houver.>

3.4 Fluxos de Eventos

3.4.1 Fluxo Básico (*Happy Day*)

<Lista de passos numerados do fluxo básico.>

1. *<Passo 1>*
2. *<Passo 2>*
3. *...*

3.4.2 Fluxos Alternativos

<Lista de fluxos alternativos (e de exceções) em relação ao fluxo básico.>

1. *< Fluxo Alternativo 1>*
2. *< Fluxo Alternativo 2>*
3. *...*

3.5 Precondições

<Lista de precondições que devem ser satisfeitas, quando houver.>

3.6 Pós-condições

<Lista de pós-condições que devem ser satisfeitas, quando houver.>

3.7 Relacionamento com outros casos de uso

<Lista de relacionamento (dependência de dados) com outros casos de uso, quando se aplicar Identifique os relacionamentos de inclusão, de extensão, e de generalização.>

3.8 Pontos de Extensão

<Lista dos pontos de extensão do caso de uso.>

3.9 Requisitos especiais

<Lista de requisitos especiais (não-funcionais) específicos do caso de uso. Exemplo: tempo de resposta.>

3.10 Regras de Negócio

<Lista das regras de negócio do caso de uso: regra 1, regra 2, ..., regra n.>

Regra 1

Nome da Regra: *<Incluir o nome da regra>*

Descrição da Regra *<Descrever a regra>*

3.11 Diagrama do caso de uso

<Modele o Diagrama do caso de uso.>

4 CENÁRIOS

<Identificar os possíveis cenários do caso de uso. Começando pelo fluxo básico e depois combinando esse fluxo com os fluxos alternativos, quando for o caso. Veja o exemplo abaixo.>

ID	Nome	Possíveis caminhos do caso de uso		
C1	*<Nome do Cenário 1>*	*Fluxo Básico*		
C2	*<Nome do Cenário 2>*	*Fluxo Básico*	*Fluxo Alternativo 1*	
C3	*<Nome do Cenário 3>*	*Fluxo Básico*	*Fluxo Alternativo 1*	*Fluxo Alternativo 2*
...	...	...	...	...
Cn	*<Nome do Cenário n>*	*Fluxo Básico*	*Fluxo Alternativo 3*	

APÊNDICE B

Guia para Elaborar a Especificação de Caso de Uso

Guia para Elaborar a Especificação de Caso de Uso

Esse guia é uma orientação de como preencher uma *Especificação de Casos de Uso* (Apêndice A). O objetivo principal é apresentar como cada seção da Especificação de Casos de Uso deve ser descrita; desta maneira, pretende-se chegar a uma "diretriz básica" de preenchimento entre os analistas de sistemas e desenvolvedores de uma organização. As seções a serem preenchidas da Especificação de Caso de Uso devem as que se seguem.

1 HISTÓRICO

Informar a data (*dd/mm/aaaa*), a versão (*X.Y*), a descrição, e o responsável pela alteração deste documento, compondo histórico de todas as alterações realizadas no documento, desde a sua formatação até sua versão mais atualizada.

2 INTRODUÇÃO

2.1 Objetivos

Especifique os objetivos (geral e específicos) do caso de uso.

2.2 Público alvo

Identifique o público alvo deste documento

2.3 Glossário

Faça as definições de todos os termos necessários a uma interpretação adequada deste documento.

2.4 Referências

Referencie todos os documentos mencionados ao longo deste documento.

3 CASO DE USO

3.1 Nome do caso de uso

O nome do caso de uso deve identificar unicamente um caso de uso. O nome dado será exclusivo, intuitivo e explicativo a fim de definir com clareza e sem ambigüidade o caso de uso. Uma boa verificação para o nome do caso de uso é pesquisar se todos os clientes, analistas e desenvolvedores compreendem os nomes e as descrições dos casos de uso.

Deve ser considerado (RUP, 2003):

- usar formato de verbo (no infinitivo) + objeto;

- cada caso de uso deve ter um nome que indica o que é alcançado pela interação com o ator. Talvez o nome precise ter várias palavras para ser entendido. Dois casos de uso não podem ter o mesmo nome;
- deixar claro o escopo do caso de uso;
- evitar nomes com operações de baixo nível (programação);

3.2 Descrição do caso de uso

A descrição deve transmitir brevemente a finalidade do caso de uso. Deve explicar de forma geral o funcionamento do caso de uso, contextualizando-o em relação ao sistema e a outros casos de uso. A descrição abordará uma explicação da finalidade principal, da proposição de valor e dos conceitos do caso de uso. Casos de uso de inclusão são fortes candidatos para serem referenciados, mas outros casos de uso sem relacionamentos claros podem ser citados. Um exemplo de descrição do caso de uso "Efetuar *Login*" é mostrada a seguir:

Este caso de uso se responsabiliza pela validação dos dados de *login* e senha do usuário. De acordo com o tipo de usuário cadastrado no sistema, a aplicação deverá validar o acesso às funcionalidades da aplicação, quando o usuário efetuar *login* na mesma.

3.3 Atores Envolvidos

Devem ser indicados os atores envolvidos no caso de uso a ser detalhado, pois os atores definem um conjunto coerente de papéis que os usuários do sistema podem desempenhar ao interagir com o caso de uso.

3.3.1 Diagrama de Hierarquia

Incluir um diagrama de hierarquia dos atores, caso haja hierarquia.

3.4 Fluxos de Eventos

3.4.1 Fluxo Básico

No fluxo básico deve ser descrito o que "geralmente" ocorre quando o caso de uso é executado. O fluxo básico é chamado de cenário de *Happy Day*, onde o fluxo de eventos descreve apenas o caminho feliz que não contém *bugs*, erros nem desvios: é como se estivesse no mundo perfeito (Cockburn, 2001). Somente existe um fluxo básico por caso de uso.

O início do fluxo básico deve descrever bem a condição de início e o contexto de negócio. O ator que inicia o caso de uso deve ser citado explicitamente. A seguir será mostrado um exemplo de fluxo básico do caso de uso "Efetuar *Login*".

Fluxo Básico (Happy Day)

1. O caso de uso se inicia quando o usuário solicita iniciar a aplicação;

2. A aplicação exibe a tela inicial e solicita o preenchimento dos seguintes campos:
    - *Login do usuário;
    - *Senha do usuário.
3. O usuário preenche os campos acima;
4. O usuário solicita efetuar *login* na aplicação;
5. A aplicação valida se os campos obrigatórios foram preenchidos;
6. A aplicação valida o *login* e a senha do usuário de acordo com a regra de negócio 1;
7. A aplicação verifica e disponibiliza as funcionalidades as quais o usuário tem acesso de acordo com seu perfil;
8. A aplicação permite o acesso do usuário à aplicação;
9. O caso de uso se encerra.

3.4.2 Fluxos Alternativos

Nos fluxos alternativos devem ser descritos os casos de exceção, cenários alternativos de uso, variação dos dados do caso de uso segundo alguma regra ou condição, restrições de acesso por tipos de atores ou usuários. Os fluxos alternativos listam eventos com seqüências diferentes não descritas no fluxo básico.

Os fluxos alternativos abordam o comportamento de caráter opcional ou excepcional em relação ao comportamento normal e também as variações do comportamento normal. Os fluxos alternativos podem ser considerados como "desvios" do fluxo básico, alguns dos quais voltarão ao fluxo de eventos básico e alguns finalizarão a execução do caso de uso.

O fluxo básico e os alternativos devem ser estruturados em passos e subfluxos, respectivamente. A principal meta deve ser a legibilidade do texto. Assim, um subfluxo deve ser um segmento de comportamento no caso de uso que tem uma finalidade clara, e ser "atômico" no sentido de que devem ser realizadas todas ou nenhuma das ações descritas. Deve-se evitar o uso de vários níveis de subfluxos, pois isso torna o texto mais complexo e difícil de entender. A seguir será mostrado o fluxo alternativo do caso de uso “Efetuar *Login*”.

Fluxos Alternativos

1. **Dados Obrigatórios Não Informados**
    1.1 Se no passo 5 do fluxo básico o usuário não tiver informado os dados obrigatórios, a aplicação exibe uma mensagem informando quais dados não foram preenchidos;
    1.2 O usuário confirma o recebimento da mensagem;

1.3 O caso de uso retorna ao passo 3 do fluxo básico, com os dados anteriormente preenchidos.

2. Regra de Negócio 1 Não Satisfeita

2.1 Se no passo 6 do fluxo básico a regra 1 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual a aplicação não pôde efetuar *login* (ver regra 1);

2.2 O usuário confirma o recebimento da mensagem;

2.3 O caso de uso retorna ao passo 3 do fluxo básico.

3. Erro no Acesso à Base de Dados ao Efetuar Login

3.1 Se após o passo 4 do fluxo básico ocorrer um erro de acesso à base de dados, quando a aplicação for efetuar o login, a aplicação exibe uma mensagem informando que ocorreu um erro no acesso a essa base;

3.2 O usuário confirma o recebimento da mensagem;

3.3 O caso de uso se encerra.

Algumas questões com relação à descrição dos passos dos eventos devem ser consideradas:

1. Na descrição das ações, utilizar verbos no presente tendo como sujeito o ator ou o sistema modelado, deixando claro que realiza a ação. As ações devem ser numeradas na ordem em que ocorrem;
2. Ao referenciar os passos de outro fluxo, indicar o número do passo e o nome do fluxo. Exemplo:

   O processamento retorna ao passo 4 do fluxo básico .
3. Cada passo (item) do fluxo deve descrever uma transação de negócio: ação atômica e completa. Deixar claro em quais pontos o controle passa do ator para o sistema e do sistema para o ator. Os dados informados devem ser descritos no mesmo item, incluindo dados que sejam de seleção de valores fornecidos pelo sistema (listas de escolha, por exemplo). Exemplo:

   Fluxo Básico

   1. O sistema solicita o preenchimento dos seguintes dados:

   - *Descrição da Atividade
   - *Lista de Projetos – campo de escolha múltipla fechada (valores possíveis para seleção: todos os projetos nos quais o usuário está registrado como gerente) – para cada projeto, o usuário deve preencher também:
     - Horas trabalhadas

- Horas extras

2. O gerente preenche os dados e solicita a inclusão.
3. O sistema exibe tela de confirmação dos dados.
4. O gerente confirma a inclusão.
5. O sistema valida os dados (<descrição das validações realizadas>...). Se os dados estiverem válidos, a tarefa gerencial é incluída no sistema.

Devem ser também descritos no fluxo de eventos os seguintes dados trocados entre ator e sistema:

- Dados de filtro:
  - ✓ Ao utilizar filtros em buscas, consultas e relatórios, indicar todos os parâmetros do filtro no fluxo de eventos. Identificar a obrigatoriedade de preenchimento de cada um.
- Resultado da consulta:
  - ✓ Indicar que dados são apresentados no resultado de uma busca, consulta ou relatório. No caso de retornar todos os dados da entidade ou de uma associação entre várias entidades, deixar explícito que todos os dados da(s) <entidade(s)> são apresentados.
- Dados do fluxo de alteração:
  - ✓ Indicar sempre que dados podem ser alterados, deixando claro quais dados servem apenas para visualização.
    - Se todos os dados puderem ser alterados indicar explicitamente
    - Se algum outro fluxo já mostrar os dados que podem ser alterados, fazer referência ao fluxo para evitar duplicar a descrição dos dados.
- Dados trocados com outros sistemas:
  - ✓ Listar no fluxo de eventos os dados trocados via arquivos, mensagens ou API (nesse caso, o formato da interface como layout, protocolos etc. não deve constar do fluxo de eventos; se já for de conhecimento, descrever na seção de Requisitos Especiais).
- Evitar detalhes de interface gráfica (lista, *combo*, *grid*, *checkbox*, *link*), a não ser que seja estritamente necessário.
- Descrever dados em alto nível (entidades e campos lógicos e não tabelas de BD).

  Exemplo:
  - ✓ O Gerente informa o projeto.

- ✓ O sistema obtém os dados das atividades através do projeto informado e apresenta para cada atividade:
  - o Descrição
  - o Membro de equipe
  - o Tipo de atividade
  - o Total de horas trabalhadas

- Campos obrigatórios de entrada devem ser identificados com um "*" na frente do nome do campo.
- Descrever regras de negócio e validação relacionadas às informações em questão.

  Exemplo:
  - ✓ O Membro de Equipe fornece para a atividade as seguintes informações:
    - o * Data da atividade: data válida superior à data do início da semana.
    - o * Número de horas trabalhadas: valor entre 1 e 24 inclusive.
- Não descrever formato ou tamanho dos campos de entrada.
- Indicar os campos que são de escolha fechada (apresentam uma lista de valores para seleção), identificando se a escolha é múltipla ou única e descrevendo o conjunto de valores possíveis. Se o conjunto de dados para seleção for um conjunto fixo, todos os valores devem ser listados.

  Exemplo:
  - ✓ Lista de projetos: campo de escolha múltipla fechada (valores possíveis: todos os projetos nos quais o Gerente está registrado)
  - ✓ Lista de tipo de atividades: campo de escolha fechada (valores possíveis: todos os tipos de atividades registradas para um projeto)
  - ✓ Situação: campo de escolha fechada (valores possíveis: Concluída, Em Atraso, Em Execução)

3.5 Precondições

As precondições de um caso de uso explicam o estado do sistema para que o caso de uso possa começar. Assim nessa seção deve(m) ser listada(s) a(s) precondição(ões) exigidas antes do início do caso de uso.

Um exemplo de precondição para o caso de uso "Efetuar *Login*" é que o usuário cadastrado recebeu um *login* para entrar no sistema.

3.6 Pós-condições

As pós-condições de um caso de uso listam os possíveis estados do sistema no fim do caso de uso. Assim nesta seção deve(m) ser listado(s) o estado(s) que o sistema pode apresentar após o término do caso de uso.

Um exemplo de pós-condição para o caso de uso "Efetuar *Login*" é que o usuário ao acessar o sistema terá disponibilizada tela principal de acordo com o perfil cadastrado, caso o *login* tenha sido digitado errado será disponibilizada uma tela de erro de *login*.

3.7 Relacionamento com outros casos de uso

Uma listagem do(s) relacionamento(s) com outros casos devem ser apresentados para que seja possível a verificação da(s) dependência(s) dos dado(s).

3.8 Pontos de Extensões

Se houver uma parte de um caso de uso que seja opcional ou desnecessária para entender a principal finalidade do caso de uso, esta parte poderá ser fatorada em um caso de uso adicional. O caso de uso adicional é usado como ponto de extensão.

3.9 Requisitos Especiais

A descrição de todos os requisitos do caso de uso, que não são abordados pelo fluxo de eventos, deve estar descrita nesta seção. Geralmente os requisitos especiais são requisitos não-funcionais, que influenciarão na execução dos casos de uso. Um exemplo de requisito especial para o caso de uso "Efetuar *Login*" é o tempo de resposta para efetuar o *login* do usuário que deve ser no máximo de 10 segundos.

3.10 Regras de Negócio

Nesta seção devem ser descritas as regras de negócio referentes ao caso de uso a ser detalhado, que devem ser descritas fora do fluxo de eventos, mas são referenciadas nos passos do caso de uso. Regras simples e genéricas são descritas nos próprios passos ou junto aos dados (como regras de validação de dados – datas, máscaras, cpf, etc). Um exemplo de regra de negócio do caso de uso "Efetuar Login":

Regra 1: Validação dos Campos *Login* e Senha

Para efetuar *login* na aplicação, o usuário deve ter sido previamente nela cadastrado pelo administrador. Além disso, o usuário deve informar a senha correta, de acordo com o que está cadastrado na base de dados da aplicação. Essas são as condições para que esta regra seja satisfeita.

3.11 Diagrama do caso de uso

Inserir o modelo de casos de uso para mostrar o relacionamento entre os casos de uso do sistema. Esta seção é opcional, pode não ser descrita. É interessante que seja mostrado o inter-relacionamento entre os casos de uso do sistema.

Um exemplo do diagrama de casos de uso mostrando o relacionamento do caso de uso "*Manter Atendente*", "*Manter Usuários*" e "Efetuar *Login*" com os outros casos de uso, como descrito nos apêndices C, D e E.

4 CENÁRIOS

Após cada caminho possível através dos fluxos (básico e alternativos) do caso de uso é possível identificar os diversos cenários de caso de uso. Nesta seção devem ser apresentadas seqüências específicas de ações que ilustram comportamentos do caso de uso. Um cenário descreve uma instância de um caso de uso, que é um determinado caminho concreto e "completo" realizado pelo caso de uso (RUP, 2003).

Um cenário descreve uma instância do caso de uso, envolvendo sempre o fluxo básico e um número qualquer de combinações de fluxos alternativos e fluxos de exceção, dispostos na ordem em que constituem o cenário.

O cenário pode ter um nome descritivo, não contendo detalhes de dados, que sejam entradas ou saídas do sistema. Esses dados devem estar presentes nos casos de teste, que são compostos a partir dos cenários.

Os apêndices C, D e E contêm a descrição dos atores relacionados ao caso de uso que está sendo detalhado.

Especificação de Caso de Uso: Manter Atendente

Projeto Alfa

ESPECIFICAÇÃO DE CASO DE USO

(UC_01 – MANTER ATENDENTE)

VERSÃO 1.0

Responsável: Tatiana Monteiro	**E-mail:** tatiana@email.com.br
Elaborador(es): Tatiana Monteiro	**E-mail:** tatiana@email.com.br

ÍNDICE

1 HISTÓRICO

Data	Versão	Descrição	Responsável
30/06/2005	*1.0*	*Criação do documento*	*Tatiana Monteiro*

2 INTRODUÇÃO

2.1 Objetivos

Este documento tem como objetivo detalhar o caso de uso Manter Atendente.

2.2 Público alvo

- *Engenheiros de requisitos;*
- *Analistas de Software;*
- *Projetistas de Software.*

2.3 Glossário

Verbete	Definição
" * "	Todos os campos que possuem um asterisco ao seu lado esquerdo são obrigatórios.
FB	*Fluxo Básico*
FA	*Fluxo Alternativo*

2.4 Referências

Não se aplica.

3 CASO DE USO

3.1 Manter Atendente

3.2 Descrição

Este caso de uso se responsabiliza pela inserção, alteração e exclusão de um atendente do Call Center. O atendente é a pessoa responsável por atender um chamado do cliente e tentar encontrar a solução para o problema do cliente. Todo atendente deverá ser um usuário já cadastrado no sistema.

3.3 Atores envolvidos

- ***Administrador***

  Este usuário é responsável por cadastrar os usuários do sistema e definir o perfil de cada usuário (coordenador do Call Center, atendente de suporte, atendente externo). Ele terá acesso completo ao sistema Call Center:

  - *Freqüência de utilização do sistema: na implantação do sistema e eventualmente para adaptações e ajustes;*
  - *Conhecimento técnico: informática básica com um profundo conhecimento do sistema.*

- ***Coordenador do Call Center***

  Este usuário é responsável por todas as operações cadastrais da aplicação, tendo permissão de inclusão, exclusão e alteração dos mesmos. Além disso, tem permissão para visualizar todos os relatórios da aplicação. A principal diferença entre o coordenador do Call Center e o administrador é que quem cadastra os usuários do sistema é o administrador.

  – *Freqüência periódica de utilização do sistema.*

  – *Conhecimento técnico de informática básica com profundo conhecimento do sistema.*

3.3.1 Diagrama de hierarquia

3.4 Fluxos de Eventos

3.4.1 Fluxo Básico (*Happy Day*) (FB)

1. *O caso de uso se inicia quando a Coordenador do Call Center solicita abrir a manutenção de atendentes.*
2. *A aplicação exibe uma lista contendo os dados dos atendentes armazenados.* ***(1T)***
3. *Uma vez que o coordenador do Call Center solicite executar uma das funções desejadas (inserir, alterar ou excluir atendente), um dos seguintes subfluxos é executado:*
   - *Se o coordenador do Call Center "Inserir", o subfluxo 3.4.1.1 Inserir Atendente é executado;*
   - *Se o coordenador do Call Center "Alterar", o subfluxo 3.4.1.2 Alterar Atendente é executado;*
   - *Se o coordenador do Call Center "Excluir", o subfluxo 3.4.1.3 Excluir Atendente é executado.*
4. *O caso de uso se encerra.*

3.4.1.1 Inserir Atendente

1. *Este subfluxo se inicia quando o coordenador do Call Center solicita inserir um novo atendente.*
2. *A aplicação solicita ao coordenador do Call Center o preenchimento dos seguintes campos:*
   - **Login do usuário: Campo de escolha fechado (Valores possíveis: todos os usuários cadastrados no Módulo de Call Center);* ***(1T)***
   - **Setor do usuário: Campo de escolha fechado (Valores possíveis: todos os setores cadastrados no Módulo de Call Center);* ***(1T)***
   - **Empresa do usuário: Campo de escolha fechado (Valores possíveis: todas as empresas cadastradas; o usuário pertence a uma delas);* ***(1T)***
   - **Tipo do atendente: Campo de escolha fechado (Valores possíveis: todos os tipos de atendentes cadastrados no Módulo de Call Center). Este campo determina o tipo de atendente que está sendo cadastrado. O atendente de suporte é responsável por registrar uma chamada no sistema. O atendente externo tem as mesmas características do atendente de suporte. Entretanto, não faz parte da área de suporte;* ***(1T)***
   - *Ramal: Campo de escolha fechado (Valores possíveis: todos os números dos ramais dos atendentes cadastrados no Módulo de Call Center);* ***(1T)***
   - **Recebimento de chamados via Internet: Campo de escolha fechado (Valores possíveis: Sim e Não). A opção "Sim" significa que o atendente pode receber chamados*

cadastrados via Internet. A opção "Não" significa o contrário. Por padrão, a opção "Não" deve vir selecionada;

- *Número dos Recebimentos de chamados via Telefone: (Valores possíveis: todos os números recebidos por cada atendente cadastrado no Módulo de Call Center);* ***(1T)***
- *Status do atendente: Campo de escolha fechado (Valores possíveis: Ativo e Inativo). A opção "Ativo" significa que o atendente está apto a atender chamados. A opção "Inativo" significa o contrário;*
- *Data de Ativação: Data em que o atendente foi cadastrado no sistema;*
- *Data de Desativação: Data em que o atendente foi desativado do sistema. Se esse campo não estiver preenchido significa que o usuário está ativo. Caso contrário, se estiver preenchido significa que o usuário está inativo. Quando um usuário está ativo significa que está apto a acessar o sistema. Do contrário, se ele está inativo, não pode acessar o sistema.*
- **Valor da hora trabalhada: Valor que o atendente recebe por cada hora trabalhada.*

3. *O coordenador do Call Center preenche os campos acima.*
4. *O coordenador do Call Center solicita salvar o cadastro do atendente.*
5. *A aplicação valida se os campos obrigatórios foram preenchidos.*
6. *A aplicação valida o login do usuário de acordo com a regra de negócio 1 (a Regra 1 é satisfeita).* ***(1T)***
7. *A aplicação realiza a inclusão dos dados informados no passo 3.* ***(1T)***
8. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do atendente foi salvo com sucesso.*
9. *O coordenador do Call Center confirma o recebimento da mensagem.*
10. *O caso de uso retorna para o passo 2 do fluxo básico.*

3.4.1.2 Alterar Atendente

1. *Este subfluxo se inicia quando o coordenador do Call Center solicita alterar um atendente selecionado.*
2. *A aplicação exibe todos os dados do atendente selecionado e permite ao coordenador do Call Center alterar qualquer um dos dados exibidos (ver lista com dados no passo 2 do subfluxo 3.4.1.1 Inserir Atendente).*
3. *O coordenador do Call Center altera os dados desejados.*
4. *O coordenador do Call Center solicita salvar o cadastro do atendente alterado.*
5. *A aplicação valida se os campos obrigatórios foram preenchidos.*
6. *A aplicação valida o login do usuário de acordo com a regra de negócio 1 (a Regra 1 é satisfeita).*
7. *A aplicação realiza a alteração dos dados modificados no passo 3.* ***(1T)***
8. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do atendente foi alterado com sucesso.*
9. *O coordenador do Call Center confirma o recebimento da mensagem.*
10. *O caso de uso retorna para o passo 2 do fluxo básico.*

3.4.1.3 Excluir Atendente

1. *Este subfluxo se inicia quando o coordenador do Call Center solicita excluir um atendente selecionado.*
2. *A aplicação exibe uma mensagem alertando o coordenador do Call Center que todos os dados do atendente em questão serão removidos.*
3. *O coordenador do Call Center confirma a exclusão.*
4. *A aplicação valida a exclusão do atendente selecionado de acordo com a regra de negócio 2 (a Regra 2 é satisfeita).* ***(1T)***

5. *A aplicação exclui o atendente da base de dados. **(1T)***
6. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do atendente foi excluído com sucesso.*
7. *O coordenador do Call Center confirma o recebimento da mensagem.*
8. *O caso de uso retorna para o passo 2 do fluxo básico.*

3.4.2 Fluxos Alternativos (FA) *(1T)*

3.4.2.1 Cancelar Manutenção de Atendentes

1. *Se no passo 3 do fluxo básico o coordenador do Call Center solicitar cancelar a manutenção de atendente, o caso de uso se encerra.*

3.4.2.2 Cancelar Inserção do Atendente

1. *Se no passo 3 do subfluxo 3.4.1.1 Inserir Atendente o coordenador do Call Center solicitar cancelar a inserção do atendente, o caso de uso retorna para o passo 1 do fluxo básico.*

3.4.2.3 Cancelar Alteração do Atendente

1. *Se no passo 3 do subfluxo 3.4.1.2 Alterar Atendente o coordenador do Call Center solicitar cancelar a alteração do atendente, o caso de uso retorna para o passo 2 do fluxo básico.*

3.4.2.4 Cancelar Exclusão do Atendente

1. *Se no passo 3 do subfluxo 3.4.1.3 Excluir Atendente o coordenador do Call Center solicitar cancelar a exclusão do atendente, o caso de uso retorna para o passo 2 do fluxo básico.*

3.4.2.5 Dados Obrigatórios Não Informados

1. *Se no passo 5 do subfluxo 3.4.1.1 Inserir Atendente ou no passo 5 do subfluxo 3.4.1.2 Alterar Atendente o coordenador do Call Center não tiver informado os dados obrigatórios, a aplicação exibe uma mensagem informando quais dados não foram preenchidos.*
2. *O coordenador do Call Center confirma o recebimento da mensagem.*
3. *A aplicação retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Atendente ou ao passo 3 do subfluxo 3.4.1.2 Alterar Atendente, respectivamente, com os dados do cadastro recém-configurado.*

3.4.2.6 Regra de Negócio 1 Não Satisfeita

1. *Se no passo 6 do subfluxo 3.4.1.1 Inserir Atendente ou no passo 6 do subfluxo 3.4.1.2 Alterar Atendente a Regra 1 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o atendente não pôde ser salvo na base de dados (ver Regra 1).*
2. *O coordenador do Call Center confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Atendente ou ao passo 3 do subfluxo 3.4.1.2 Alterar Atendente, respectivamente, com os dados do cadastro recém-configurado.*

3.4.2.7 Regra de Negócio 2 Não Satisfeita

1. *Se no passo 4 do subfluxo 3.4.1.3 Excluir Atendente a Regra 2 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o atendente selecionado não pôde ser excluído da base de dados (ver Regra 2).*
2. *O coordenador do Call Center confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 2 do fluxo básico.*

3.4.2.8 Erro no Acesso à Base de Dados ao Exibir Dados dos Atendentes

1. *Se no passo 2 do fluxo básico ocorrer um erro de acesso à base de dados, quando a aplicação for exibir a lista contendo os dados dos atendentes, a aplicação exibe uma mensagem informando que ocorreu um erro no acesso a base de dados.*
2. *O coordenador do Call Center confirma o recebimento da mensagem.*
3. *O caso de uso se encerra.*

3.5 Precondições

- *O coordenador do Call Center deverá ter efetuado login no sistema para poder executar esta funcionalidade.*

3.6 Pós-Condições

- *O atendente inserido ou alterado pelo coordenador do Call Center deverá ser salvo na base de dados;*
- *O atendente excluído pelo coordenador do Call Center deverá ser removido da base de dados.*

3.7 Relacionamento com outros casos de uso

- *<Inclui>: Efetuar Login*

3.8 Pontos de Extensão

Não se aplica.

3.9 Requisitos especiais

Não se aplica.

3.10 Regras de Negócio

Regra 1

Nome da Regra: *Existência de atendente com o Mesmo Login*

Descrição da Regra:

Não é permitido cadastrar um atendente mais de uma vez na aplicação. Essa é a condição para que a regra seja satisfeita.

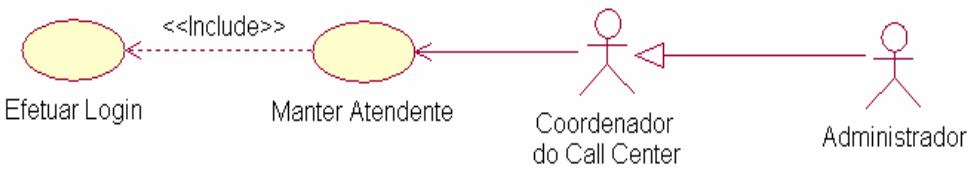
Regra 2

Nome da Regra: *Proibição de Exclusão do Atendente*

Descrição da Regra:

Um atendente só pode ser excluído do sistema se ele não estiver associado a nenhuma outra funcionalidade da aplicação. Ou seja, se existir um chamado ou outra entidade do sistema relacionada de alguma forma com o atendente em questão, este atendente não poderá ser excluído. Essa é a condição para que referida regra seja satisfeita.

3.11 Diagrama do caso de uso



4 CENÁRIOS

ID	Nome	Possíveis caminhos do caso de uso		
C1	***Fluxo Básico – Inserir Atendente***	***FB – 3.4.1***	***FB – 3.4.1.1***	
C2	*Inserir Atendente – Cancelar manutenção de atendentes*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.1*
C3	*Inserir Atendente – Cancelar inserção do atendente*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.2*
C4	*Inserir Atendente – Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.5*
C5	*Inserir Atendente – Regra de negócio 1 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.6*
C6	*Inserir Atendente – Erro de acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.8*
C7	***Fluxo Básico – Alterar Atendente***	***FB – 3.4.1***	***FB – 3.4.1.2***	
C8	*Alterar Atendente - Cancelar manutenção de atendentes*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.1*
C9	*Alterar Atendente - Cancelar alteração do atendente*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.3*
C10	*Alterar Atendente - Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.5*
C11	*Alterar Atendente - Regra de negócio 1 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.6*
C12	*Alterar Atendente - Erro de acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.8*
C13	***Fluxo Básico – Excluir Atendente***	***FB – 3.4.1***	***FB – 3.4.1.3***	
C14	*Excluir Atendente - Cancelar manutenção de atendentes*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.1*
C15	*Excluir Atendente - Cancelar exclusão do atendente*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.4*
C16	*Excluir Atendente - Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.5*
C17	*Excluir Atendente - Regra de negócio 2 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.7*
C18	*Excluir Atendente - Erro de acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.8*

APÊNDICE D

Especificação de Caso de Uso: Manter Usuários

Projeto Alfa

ESPECIFICAÇÃO DE CASO DE USO

(UC_02 - MANTER USUÁRIOS)

VERSÃO 1.0

Responsável: Tatiana Monteiro	**E-mail:** tatiana@email.com.br
Elaborador(es): Tatiana Monteiro	**E-mail:** tatiana@email.com.br

ÍNDICE

1 HISTÓRICO

Data	Versão	Descrição	Responsável
30/06/2005	*1.0*	*Criação do documento*	*Tatiana Monteiro*

2 INTRODUÇÃO

2.1 Objetivos

Este documento tem como objetivo detalhar o caso de uso Manter Usuários.

2.2 Público alvo

- *Engenheiros de requisitos;*
- *Analistas de Software;*
- *Projetistas de Software.*

2.3 Glossário

Verbete	Definição
" * "	Todos os campos que possuem um asterisco ao seu lado esquerdo são obrigatórios.
FB	*Fluxo Básico*
FA	*Fluxo Alternativo*

2.4 Referências

Não se aplica.

3 CASO DE USO

3.1 Manter Usuários

3.2 Descrição

Este caso de uso se responsabiliza pela inserção, alteração e exclusão de um usuário do sistema Call Center. Os tipos de usuário do Sistema Call Center são: Administrador, Coordenador de Suporte, e Usuários (Consultor de Suporte, Consultor Interno, e Consultor Externo).

3.3 Atores envolvidos

- ***Administrador***

  Este ator é responsável por cadastrar os usuários do sistema e definir o perfil de cada usuário (coordenador de suporte, consultor de suporte, consultor externo). Ele terá acesso completo ao sistema Call Center:

  - *Freqüência de utilização do sistema na implantação do sistema e eventualmente para adaptações e ajustes;*
  - *Conhecimento técnico em Informática básica com profundo conhecimento do sistema.*

- ***Coordenador de Suporte***

  Este ator é responsável por todas as operações cadastrais da aplicação, tendo permissão de inclusão, exclusão e alteração dos mesmos. Além disso, tem permissão para visualizar todos os relatórios da aplicação. A principal diferença entre o coordenador de suporte e o administrador é que quem cadastra os usuários do sistema é o administrador.

  - *Freqüência de utilização do sistema: periódica.*
  - *Conhecimento técnico em Informática básica com profundo conhecimento do sistema.*
- ***Usuário***

  Este ator é quem utiliza operacionalmente o sistema.

3.3.1 Diagrama de hierarquia

3.4 Fluxos de Eventos

3.4.1 Fluxo Básico (*Happy Day*) (FB)

1. *O caso de uso se inicia quando o administrador solicita abrir o cadastro de usuários.*
2. *A aplicação exibe uma lista com as seguintes informações dos usuários cadastrados no sistema:*
   - **Nome do usuário;*
   - **Login do usuário;*
   - **Perfil do usuário;*
   - **Status do usuário.*
3. *A aplicação exibe uma lista contendo os dados dos usuários armazenados.*
4. *Uma vez que o administrador solicite executar uma das funções desejadas (inserir, alterar ou excluir usuário), um dos seguintes subfluxos é executado:*
   - *Se o administrador solicitar "Inserir", o subfluxo 3.4.1.1 Inserir Usuário é executado;*
   - *Se o administrador solicitar "Alterar", o subfluxo 3.4.1.2 Alterar Usuário é executado;*
   - *Se o administrador solicitar "Excluir", o subfluxo 3.4.1.3 Excluir Usuário é executado.*
5. *O caso de uso se encerra.*

3.4.1.1 Inserir Usuário

1. *Este subfluxo se inicia quando o administrador solicita inserir um novo usuário.*
2. *A aplicação solicita ao administrador o preenchimento dos seguintes campos:*
   - **Nome do usuário;*
   - **Login do usuário;*
   - **Senha do usuário;*
   - **Perfil do usuário: Campo de escolha fechado (Valores possíveis: Administrador, Coordenador de Suporte, Consultor de Suporte e Consultor Externo);*
   - **Data de Ativação: Data em que o usuário foi cadastrado no sistema;*
   - *Data de Desativação: Data em que o usuário foi desativado do sistema. Se esse campo não estiver preenchido significa que o usuário está ativo. Caso contrário, se estiver preenchido significa que o usuário está inativo.Um usuário está ativo quando está apto a acessar o sistema. Do contrário, se ele está inativo, não pode acessar o sistema.*
3. *O administrador preenche os campos acima.*
4. *O administrador solicita salvar o cadastro do usuário.*
5. *A aplicação valida se os campos obrigatórios foram preenchidos.*

6. *A aplicação valida o login do usuário de acordo com a regra de negócio 1 (a Regra 1 é satisfeita).*
7. *A aplicação valida a senha do usuário de acordo com a regra de negócio 2 (a Regra 2 é satisfeita).*
8. *A aplicação valida os campos data de ativação e data de desativação de acordo com a regra de negócio 3 (a Regra 3 é satisfeita).*
9. *A aplicação realiza a inclusão dos dados informados pelo administrador no passo 3.*
10. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do usuário foi salvo com sucesso.*
11. *O administrador confirma o recebimento da mensagem.*
12. *O caso de uso retorna para o passo □3 do fluxo básico.*

3.4.1.2 Alterar Usuário

1. *Este subfluxo se inicia quando o administrador solicita alterar um usuário selecionado.*
2. *A aplicação exibe todos os dados do usuário selecionado e permite ao administrador alterar qualquer um dos dados exibidos, com exceção do nome do usuário (ver lista com dados no passo 2 do subfluxo 3.4.1.1 Inserir Usuário).*
3. *O administrador altera os dados desejados.*
4. *O administrador solicita salvar o cadastro do usuário alterado.*
5. *A aplicação valida se os campos obrigatórios foram preenchidos.*
6. *A aplicação valida o login do usuário de acordo com a regra de negócio 1 (a Regra 1 é satisfeita).*
7. *A aplicação valida a senha do usuário de acordo com a regra de negócio 2 (a Regra 2 é satisfeita).*
8. *A aplicação valida os campos data de ativação e data de desativação de acordo com a regra de negócio 3 (a Regra 3 é satisfeita).*
9. *A aplicação realiza a alteração dos dados modificados pelo administrador no passo 3.*
10. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do usuário foi alterado com sucesso.*
11. *O administrador confirma o recebimento da mensagem.*
12. *O caso de uso retorna para o passo □3 do fluxo básico.*

3.4.1.3 Excluir Usuário

1. *Este subfluxo se inicia quando o administrador solicita excluir um usuário selecionado.*
2. *A aplicação exibe uma mensagem alertando o administrador que todos os dados do usuário em questão serão removidos.*
3. *O administrador confirma a exclusão.*
4. *A aplicação valida a exclusão do usuário selecionado de acordo com a regra de negócio 4 (a Regra 4 é satisfeita).*
5. *A aplicação exclui o usuário da base de dados.*
6. *A aplicação exibe uma mensagem de confirmação informando que o cadastro do usuário foi excluído com sucesso.*
7. *O administrador confirma o recebimento da mensagem.*
8. *O caso de uso retorna para o passo □3 do fluxo básico.*

3.4.2 Fluxos Alternativos (FA)

3.4.2.1 Dados Obrigatórios Não Informados

1. *Se no passo 5 do subfluxo 3.4.1.1 Inserir Usuário ou no passo 5 do subfluxo 3.4.1.2 Alterar Usuário o administrador não tiver informado os dados obrigatórios, a aplicação exibe uma mensagem informando quais dados não foram preenchidos.*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Usuário ou ao passo 3 do subfluxo 3.4.1.2 Alterar Usuário, respectivamente, com os dados do cadastro recém configurado.*

3.4.2.2 Regra de Negócio 1 Não Satisfeita

1. *Se no passo 6 do subfluxo 3.4.1.1 Inserir Usuário ou no passo 6 do subfluxo 3.4.1.2 Alterar Usuário a Regra 1 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o usuário não pôde ser salvo na base de dados (ver Regra 1).*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Usuário ou ao passo 3 do subfluxo 3.4.1.2 Alterar Usuário, respectivamente, com os dados do cadastro recém-configurado.*

3.4.2.3 Regra de Negócio 2 Não Satisfeita

1. *Se no passo 7 do subfluxo 3.4.1.1 Inserir Usuário ou no passo 7 do subfluxo 3.4.1.2 Alterar Usuário a Regra 2 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o usuário não pôde ser salvo na base de dados (ver Regra 2).*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Usuário ou ao passo 3 do subfluxo 3.4.1.2 Alterar Usuário, respectivamente, com os dados do cadastro recém-configurado.*

3.4.2.4 Regra de Negócio 3 Não Satisfeita

1. *Se no passo 8 do subfluxo 3.4.1.1 Inserir Usuário ou no passo 8 do subfluxo 3.4.1.2 Alterar Usuário a Regra 3 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o usuário não pôde ser salvo na base de dados (ver Regra 3).*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do subfluxo 3.4.1.1 Inserir Usuário ou ao passo 3 do subfluxo 3.4.1.2 Alterar Usuário, respectivamente, com os dados do cadastro recém-configurado.*

3.4.2.5 Regra de Negócio 4 Não Satisfeita

1. *Se no passo 4 do subfluxo 3.4.1.3 Excluir Usuário a Regra 4 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual o usuário selecionado não pôde ser excluído da base de dados (ver Regra 4).*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso retorna ao passo 3 do fluxo básico.*

3.4.2.6 Erro no Acesso à Base de Dados ao Exibir Dados dos Usuários

1. *Se no passo □3 do fluxo básico ocorrer um erro de acesso à base de dados, quando a aplicação for exibir a lista contendo os dados dos usuários, a aplicação exibe uma mensagem informando que ocorreu um erro no acesso a base de dados.*
2. *O administrador confirma o recebimento da mensagem.*
3. *O caso de uso se encerra.*

3.5 Precondições

- *O administrador deverá ter efetuado login no sistema para poder executar esta funcionalidade.*

3.6 Pós-Condições

- *O usuário inserido ou alterado pelo administrador deverá ser salvo na base de dados;*
- *O usuário excluído pelo administrador deverá ser removido da base de dados.*

3.7 Relacionamento com outros casos de uso

- *<Inclui>: Efetuar Login*

3.8 Pontos de Extensão

Não se aplica.

3.9 Requisitos Especiais

Não se aplica.

3.10 Regras de Negócio

Regra 1

Nome da Regra: *Existência de um usuário com o mesmo login*

Descrição da Regra

Não é permitido cadastrar mais de um usuário com um mesmo login na aplicação. Essa é a condição para que a regra seja satisfeita.

Regra 2

Nome da Regra: *Validação do campo Senha*

Descrição da Regra

O campo senha deverá conter no mínimo cinco e no máximo 16 caracteres, como condição para que esta regra seja satisfeita.

Regra 3

Nome da Regra: *Validação dos Campos Data de Ativação e Data de Desativação*

Descrição da Regra

As seguintes condições deverão ser verdadeiras para que essa regra seja satisfeita:

- *Os campos "Data de Ativação" e "Data de Desativação" deverão seguir o padrão dd/mm/aaaa;*
- *A data de ativação não poderá ser menor que a data atual do servidor;*
- *A data de desativação deverá ser maior que a data de ativação.*

Se alguma das condições acima não for verdadeira, esta regra não será satisfeita.

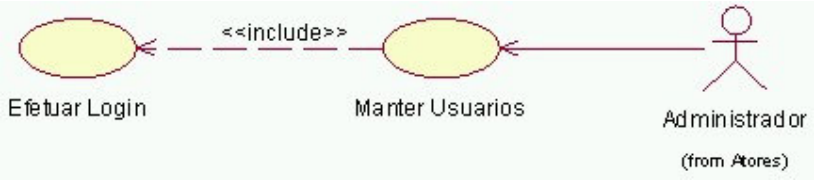
Regra 4

Nome da Regra: *Proibição de Exclusão do Usuário*

Descrição da Regra

Um usuário só pode ser excluído do sistema se não estiver associado a nenhuma outra funcionalidade da aplicação. Ou seja, se existir um chamado ou outra entidade do sistema relacionada de alguma forma com o usuário em questão, este usuário não poderá ser excluído da base de dados. Essa é a condição para que referida regra seja satisfeita.

3.11 Diagrama do caso de uso



4 CENÁRIOS

ID	Nome	Possíveis caminhos do caso de uso		
C1	***Fluxo Básico – Inserir Usuário***	***FB – 3.4.1***	***FB – 3.4.1.1***	
C2	*Inserir Usuário – Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.1*
C3	*Inserir Usuário – Regra de negócio 1 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.2*
C4	*Inserir Usuário - Regra de negócio 2 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.3*
C5	*Inserir Usuário - Regra de negócio 3 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.4*
C6	*Inserir Usuário – Erro no acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.1*	*FA – 3.4.2.6*
C7	***Fluxo Básico – Alterar Usuário***	***FB – 3.4.1***	***FB – 3.4.1.2***	
C8	*Alterar Usuário – Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.1*
C9	*Alterar Usuário - Regra de negócio 1 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.2*
C10	*Alterar Usuário - Regra de negócio 2 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.3*
C11	*Alterar Usuário - Regra de negócio 3 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.4*
C12	*Alterar Usuário - Erro no acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.2*	*FA – 3.4.2.6*
C13	***Fluxo Básico – Excluir Usuário***	***FB – 3.4.1***	***FB – 3.4.1.3***	
C14	*Excluir Usuário - Dados obrigatórios não informados*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.3.1*
C15	*Excluir Usuário - Regra de negócio 4 não satisfeita*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.3.5*
C16	*Excluir Usuário - Erro no acesso à base de dados*	*FB – 3.4.1*	*FB – 3.4.1.3*	*FA – 3.4.2.6*

Especificação de Caso de Uso: Efetuar *Login*

Projeto Alfa

ESPECIFICAÇÃO DE CASO DE USO

(UC_03 - EFETUAR *LOGIN*)

VERSÃO 1.0

Responsável: Tatiana Monteiro	**E-mail:** tatiana@email.com.br
Elaborador(es): Tatiana Monteiro	**E-mail:** tatiana@email.com.br

ÍNDICE

1 HISTÓRICO

Data	Versão	Descrição	Responsável
30/06/2005	*1.0*	*Criação do documento*	*Tatiana Monteiro*

2 INTRODUÇÃO

2.1 Objetivos

Este documento tem como objetivo detalhar o caso de uso Efetuar Login

2.2 Público alvo

- *Engenheiros de requisitos;*
- *Analistas de Software;*
- *Projetistas de Software.*

2.3 Glossário

Verbete	Definição
" * "	Todos os campos que possuem um asterisco ao seu lado esquerdo são obrigatórios.
FB	*Fluxo Básico*
FA	*Fluxo Alternativo*

2.4 Referências

Não se aplica.

3 CASO DE USO

3.1 Efetuar *Login*

3.2 Descrição

Este caso de uso se responsabiliza pela validação dos dados de login e senha do usuário. De acordo com o tipo de usuário cadastrado no sistema, a aplicação deverá validar o acesso às funcionalidades da aplicação, quando o usuário efetuar login na mesma.

3.3 Atores envolvidos

- ***Administrador***

  Este ator é responsável por cadastrar os usuários do sistema e definir o perfil de cada usuário (coordenador de suporte, consultor de suporte, consultor externo). Ele terá acesso completo ao sistema Call Center:

  - *Freqüência de utilização do sistema: na implantação do sistema e eventualmente para adaptações e ajustes;*
  - *Conhecimento técnico em Informática básica com profundo conhecimento do sistema.*

- ***Coordenador de Suporte***

  Este ator é responsável por todas as operações cadastrais da aplicação, tendo permissão de inclusão, exclusão e alteração dos mesmos. Além disso, tem permissão para visualizar todos os relatórios da aplicação. A principal diferença entre o coordenador de suporte e o administrador é que quem cadastra os usuários do sistema é o administrador.

  – *Freqüência de utilização do sistema: periódica.*

  – *Conhecimento técnico em Informática básica com profundo conhecimento do sistema.*

- ***Usuário***

  Este ator é quem utiliza operacionalmente o sistema.

3.3.1 Diagrama de hierarquia

3.4 Fluxos de Eventos

3.4.1 Fluxo Básico (*Happy Day*) (FB)

1. *O caso de uso se inicia quando o usuário solicita iniciar a aplicação.*
2. *A aplicação exibe a tela inicial e solicita o preenchimento dos seguintes campos:*
   - **Login do usuário;*
   - **Senha do usuário.*
3. *O usuário preenche os campos acima.*
4. *O usuário solicita efetuar login na aplicação.*
5. *A aplicação valida se os campos obrigatórios foram preenchidos.*
6. *A aplicação valida o login e a senha do usuário de acordo com a regra de negócio 1 (a regra 1 é satisfeita).*
7. *A aplicação verifica e disponibiliza as funcionalidades as quais o usuário tem acesso de acordo com seu perfil.*
8. *A aplicação permite o acesso do usuário à aplicação.*
9. *O caso de uso se encerra.*

3.4.2 Fluxos Alternativos (FA)

1. Dados Obrigatórios Não Informados

   1.1 Se no passo 5 do fluxo básico o usuário não tiver informado os dados obrigatórios, a aplicação exibe uma mensagem informando quais dados não foram preenchidos.

   1.2 O usuário confirma o recebimento da mensagem.

   1.3 O caso de uso retorna ao passo 3 do fluxo básico, com os dados anteriormente preenchidos.

2. Regra de Negócio 1 Não Satisfeita

   2.1 Se no passo 6 do fluxo básico a Regra 1 não for satisfeita, a aplicação exibe uma mensagem informando o motivo pelo qual a aplicação não pôde efetuar login (ver regra 1).

   2.2 O usuário confirma o recebimento da mensagem.

   2.3 O caso de uso retorna ao passo 3 do fluxo básico.

3. Erro no Acesso à Base de Dados ao Efetuar Login

3.1 Se após o passo 4 do fluxo básico ocorrer um erro de acesso à base de dados, quando a aplicação for efetuar o login, a aplicação exibe uma mensagem informando que ocorreu um erro no acesso à base de dados.

3.2 O usuário confirma o recebimento da mensagem.

3.3 O caso de uso se encerra.

3.5 Precondições

- *O usuário esteja cadastrado no sistema.*

3.6 Pós-Condições

- *O usuário ao se conectar ao sistema terá disponibilizada tela principal de acordo com o perfil cadastrado.*

3.7 Relacionamento com outros casos de uso

Não se aplica.

3.8 Pontos de Extensão

Não se aplica.

3.9 Requisitos especiais

O tempo de resposta para efetuar o login do usuário deve ser no máximo de 10 segundos.

3.10 Regras de Negócio

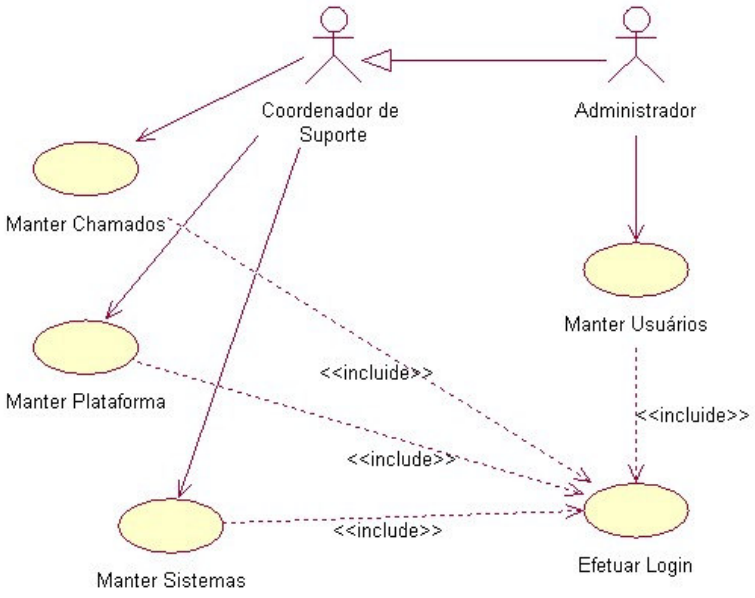
Regra 1

Nome da Regra: *Validação dos Campos Login e Senha*

Descrição da Regra

Para efetuar login na aplicação, o usuário deve ter sido previamente cadastrado na aplicação pelo administrador. Além disso, o usuário deve informar a senha correta, de acordo com o que está cadastrado na base de dados da aplicação. Estas são as condições para que referida regra seja satisfeita.

3.11 Diagrama do caso de uso



4 CENÁRIOS

ID	Nome	Possíveis caminhos do caso de uso	
C1	*Happy Day – Efetuar Login*	*FB*	
C2	*Dados Obrigatórios Não Informados*	*FB*	*FA – 1*
C3	*Regra de Negócio 1 Não Satisfeita*	*FB*	*FA – 2*
C4	*Erro no Acesso à Base de Dados ao Efetuar Login*	*FB*	*FA – 3*

Modelo de Regras de Negócio

<Nome do Projeto>

REGRAS DE NEGÓCIO

Versão <X.Y>

Responsável:	**E-mail:**
Elaborador(es):	**E-mail:**

ÍNDICE

1 HISTÓRICO

Data	Versão	Descrição	Responsável
<dd/mm/aaaa>	*<X.Y>*	*<detalhes>*	*<nome>*

2 INTRODUÇÃO

2.1 Objetivos

<Descrever as principais regras do negócio relativo ao projeto <nome do projeto>. Estas regras poderão ser referenciadas em um ou mais casos de uso.>

2.2 Público alvo deste documento

- *<Lista do publico alvo do documento.>*

2.3 Glossário

<Listar o glossário relacionado a este documento.>

2.4 Referências

[1] <Ajustar conforme o documento>.

3 DEFINIÇÕES DAS REGRAS DE NEGÓCIO

3.1 <RN_nº da regra de negócio> - <Nome da Regra de Negócio>

<Incluir as regras de negócio do projeto. Para cada regra deverá ser criada uma seção a ela referente.>