Teoria Explanatória para Estimativa Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos

Everton Luiz Vieira
UFSC-CTC-INE - Universidade Federal de Santa Catarina
Florianópolis, Brasil, 88040-900
evertonv@inf.ufsc.br

e

Raul Sidnei Wazlawick
UFSC-CTC-INE - Universidade Federal de Santa Catarina
Florianópolis, Brasil, 88040-900
raul@inf.ufsc.br

Abstract

This paper presents an explanation on the Use Case Points (UCP) method for software effort estimation. Basically, the paper develops a theory that seeks to explain why the technique based on Mandatory Steps developed by the authors tend to produce better estimation than the technique based on straight counting of steps and their multiplication by environmental factors. The paper presents a theoretic analysis on the main factors that produce development effort on an object oriented development process. It separates effort estimation into factors that are dependent or independent on the complexity of the system. The main contribution of this paper is the explanation on the interference of dependent and independent factors in object oriented developing, because the original UCP method was developed as an evolution of the Function Points that was originally proposed for structured software development. The paper shows that mandatory steps have a strong influence on effort and that environmental factors may not be dependent on the complexity of the system as in the original UCP method.

Keywords: Use Case Points, Metrics, Measurement.

Resumo

Este artigo apresenta uma explanação sobre o funcionamento do método de estimativa Use Case Points (UCP) ou Pontos de Caso de Uso para o tempo de desenvolvimento de software orientado a objetos, separando as estimativas de esforço em dependentes e independentes da complexidade do sistema. Procura-se explicar porque a técnica de medição baseada em Passos Obrigatórios, desenvolvida pelos autores, tende a produzir melhores estimativas do que a técnica baseada na classificação de casos de uso e atores em simples, médio e complexo, com pesos multiplicados pelos fatores ambientais. A principal contribuição do artigo consiste em explicar a interferência dos fatores dependentes e independentes no desenvolvimento orientado a objetos, visto que o método de pontos de caso de uso foi desenvolvido como uma evolução do método de Pontos de Função, o qual foi originalmente proposto para o desenvolvimento estruturado. O artigo mostra que os passos obrigatórios dos casos de uso no desenvolvimento orientado a objetos têm forte influência no valor final do esforço despendido devido ao encadeamento entre os artefatos produzidos, e que fatores ambientais podem não ser dependentes da complexidade do sistema, como no método de UCP original.

Palavras chaves: Pontos de Caso de Uso, Métricas, Medição.

1. INTRODUÇÃO

O método de pontos de caso de uso foi proposto inicialmente [6][7] como uma adaptação do já consagrado método de pontos de função [1] [15] para aplicação ao projeto orientado a objetos. O método, porém, tem ainda várias limitações [5] [9] [11] [13] [14]. A falta de um padrão para escrever casos de uso e a diversidade de formas de contagem do número de passos tende a produzir resultados muito discrepantes [3] [16]. Este artigo trabalha com a hipótese de que o método de contagem que classifica atores e casos de uso em simples, médios e complexos, e que multiplica estes fatores de complexidade por fatores ambientais, tem limitações para aplicação ao projeto orientado a objetos, mas que podem ser superadas.

O artigo apresenta um desenvolvimento teórico, que mostra como a contagem de passos nos casos de uso resulta em efetivo esforço de desenvolvimento quando se trabalha com um processo orientado a objetos. As equações resultantes podem ser aplicadas em diferentes momentos do processo de desenvolvimento e os fatores envolvidos podem ser ajustados conforme a capacidade técnica da equipe e outros fatores ambientais.

A chave para esta abordagem reside no fato de que o desenvolvimento orientado a objetos difere em essência do desenvolvimento estruturado. O artigo mostra que a partir da expansão dos casos de uso existe um determinado conjunto de passos denominados "obrigatórios", que determinam o esforço de desenvolvimento, proporcionalmente ao seu número, enquanto que outros passos, denominados "complementares", não têm influência sobre o esforço de desenvolvimento.

Já os fatores ambientais, usualmente são usados para calcular o esforço final de desenvolvimento através de uma multiplicação de seus pesos pelo esforço calculado pela complexidade dos atores e casos de uso. Porém, este artigo desenvolve uma série de equações que demonstram que no desenvolvimento orientado a objetos, estes fatores poderão ser não apenas multiplicados, mas *somados* à complexidade dos casos de uso, dependendo de sua natureza. Por exemplo, a necessidade de estudar uma determinada ferramenta de desenvolvimento ou a criação de um determinado *framework* que vai acelerar etapas posteriores do desenvolvimento, usualmente não dependem da complexidade do sistema a ser desenvolvido. Mais do que isso, essas atividades podem causar aumento do esforço, especificado por uma constante, que independe da complexidade final do sistema e uma diminuição no esforço, determinada por uma constante *dependente* da complexidade do sistema.

2. O MÉTODO DE DESENVOLVIMENTO E O MÉTODO DE ESTIMATIVA

O método de desenvolvimento de software orientado a objetos que serviu de base para a definição das equações de estimativa apresentadas neste artigo é o método de Larman [8] adaptado por Wazlawick [17]. Esta seção apresenta um resumo do método, relacionando as atividades e etapas relevantes com a definição de uma teoria para o cálculo do esforço de desenvolvimento de software. A medida que as etapas são apresentadas, os fatores que determinam o esforço são identificados e incorporados em equações que determinam o esforço de desenvolvimento. Essas equações, se devidamente ajustadas, poderão ser mais precisas, para a previsão de esforço no desenvolvimento orientado a objetos.

2.1 Primeira Etapa: Listagem dos Casos de Uso na Fase de Concepção

Vários processos de análise já ocorreram até que seja possível listar os casos de uso na fase de concepção [8] (como, por exemplo, o estudo de viabilidade e o levantamento de requisitos), mas para efeito de cálculo de esforço de desenvolvimento baseado em casos de uso, a primeira grandeza que pode ser medida é o número de casos de uso. Os casos de uso que compõe o sistema são listados nesta etapa e uma pequena descrição de cada um deles é apresentada. Neste ponto, a estimação de complexidade de cada caso de uso dependerá apenas da experiência do analista. É possível imaginar que casos de uso do tipo inserir/alterar/excluir/consultar/listar sejam mais simples do que outros que tratam de processos de negócio, mas dependendo da quantidade de restrições colocadas sobre estes casos de uso, eles também poderão eventualmente se tornar complexos.

Pode-se somente com esta informação, trabalhar com dois tipos de estimativa, uma que considera uma complexidade média inerente aos casos de uso e outra que utiliza a experiência do analista para classificar casos de uso de acordo com sua complexidade. Essa medida de complexidade pode ser discreta, como: simples = 1, médio = 3 e complexo = 5; ou contínua, onde o caso de uso mais simples terá valor próximo de 0, o médio, valor próximo de 1 e o complexo um valor arbitrariamente maior do que 1, dentro de alguma escala subjetiva do analista.

Para manter a teoria o mais geral possível será usada a escala contínua, pois em fase subseqüentes, ao invés da intuição do analista, poderão ser usadas informações mais precisas sobre a complexidade de cada caso de uso.

2.1.1 *Estimativa usando apenas o número de casos de uso*

A primeira técnica de estimativa possível nesta fase, pode considerar dois fatores:

- O número de casos de uso: N .
- O tempo médio para desenvolver todos os artefatos (inclusive o código) que derivam de um caso de uso: T .

O tempo médio para desenvolver todos os artefatos derivados de um caso de uso, pode ser calculado experimentalmente a partir de vários projetos, tomando-se o tempo total para desenvolver o projeto (TD) dividido pelo número de casos de uso (N). Então, será calculado como a média dos valores TD / N . A equação de estimativa de tempo pode ser, portanto, nesta fase, enunciada assim:

(1) $TDE = N * T$

Onde: TDE é o Tempo total de Desenvolvimento Estimado para o software.

Esta técnica de estimativa tende a ser pouco precisa, pois a simples média não dará bons resultados quando a complexidade dos casos de uso for variável, ou seja, espera-se que pelas características distintas dos projetos e diversidade de tipos de casos de uso, o desvio padrão desta média seja muito alto.

Sendo assim, é necessário adicionar a esta equação uma medida de complexidade para cada caso de uso.

2.1.2 Acrescentando uma medida de complexidade aos casos de uso

A segunda técnica considera uma medida subjetiva do analista para classificar a complexidade dos casos de uso. Esta técnica poderá ser mais precisa que a anterior à medida que o analista for capaz de apresentar uma estimativa de complexidade mais próxima da realidade para cada caso de uso [2]. Esta técnica aproxima-se mais da técnica de UCP, que classifica os casos de uso em simples, médio e complexo.

A variável T continua denotando o tempo médio para desenvolver um caso de uso, mas ela é agora multiplicada pelo fator de complexidade específico para cada caso de uso, definido pelo analista. Este fator é dado por uma função $complex(uc)$, a qual associa um valor racional entre 0 e infinito (possivelmente), a cada caso de uso (uc).

A equação de estimativa de esforço poderia ser, então, assim enunciada:

(2) $TDE = T\sum_{i=1}^{N} complex(uc_i)$

Para que a equação 2 seja consistente com a equação 1, deve-se presumir inicialmente que na média $\sum_{i=1}^{N} complex(uc_i) = N$, ou seja, a somatória dos valores de complexidade atribuídos a cada caso de uso em um projeto, deve ser igual ao número de casos de uso do projeto. Porém, isso só deve ser fixado desta forma no momento da determinação experimental destes valores.

Na estimação real dos projetos, a fórmula $\sum_{i=1}^{N} complex(uc_i)$ poderá ser diferente de N . Nestes casos, quanto mais negativo for o valor de $\sum_{i=1}^{N} complex(uc_i) - N$, mais simples serão os casos de uso do projeto na média, e quanto mais positivo for este valor, mais complexos serão os casos de uso do projeto na média.

2.1.3. Separando fatores dependentes e independentes dos casos de uso.

Se o analista puder estimar o esforço de desenvolvimento que não depende da quantidade de casos de uso, então a função $complex(uc)$ pode ser redefinida com mais precisão, como sendo o tempo *efetivamente* empregado no desenvolvimento dos artefatos derivados de um determinado caso de uso.

Seja TFI o tempo total estimado para atividades que independem da quantidade ou complexidade dos casos de uso, então a equação da estimativa de esforço pode ser assim redefinida:

(3) $TDE = TFI + T\sum_{i=1}^{N} complex(uc_i)$

Exemplos de atividades independentes da complexidade do sistema em desenvolvimento são: treinamento da equipe, desenvolvimento de *frameworks*, aquisição de ferramentas de desenvolvimento, etc.

2.1.4. Considerando que a análise inicial não é completa.

Outro aspecto que não pode ser negligenciado, é que o número de casos de uso N obtido no início do processo de análise, tende a ser menor do que o número real de casos de uso ($N' = f * N$) contabilizado quando o sistema estiver pronto, ou seja, o número de casos de uso tende a aumentar à medida que o processo de desenvolvimento prossegue.

O fator de aumento do número de casos de uso(f), pode nesta etapa inicial, apenas ser previsto como uma média. Este fator pode depender do tipo de sistema e também do estilo de análise.

Um analista mais despreocupado identificará apenas os principais casos de uso quando o número real é muito maior, o que determina um fator f mais alto. Um analista mais detalhista poderá identificar praticamente todos os casos de uso

na fase de concepção, não importando quão elementares eles sejam. Com isso, haverá poucos novos casos de uso a serem descobertos à medida que a análise prossegue. Neste caso, f terá um valor muito próximo de 1.
Não se trata aqui de considerar aspectos relacionados a novos requisitos, que porventura surjam durante o processo de desenvolvimento, mas do fator de imprecisão da análise que muitas vezes impede que o analista perceba na fase de concepção, a totalidade dos casos de uso que realmente serão necessários.
Aplicando-se esse fator à equação (3) obtém-se:

$$(4)\ TDE = TFI + T\sum_{i=1}^{f*N} complex(uc_i)$$

Como o valor de f consiste em um valor médio, possivelmente não inteiro, poderá ser conveniente simplificar esta equação para a seguinte forma:

$$(4')\ TDE = TFI + T * f\sum_{i=1}^{N} complex(uc_i)$$

Neste ponto da análise em que apenas a listagem dos casos de uso está disponível, um analista poderá fazer uma previsão de esforço mais precisa à medida que conhecer valores médios históricos para os elementos da equação (4'). Não sendo conhecidos alguns destes valores à fórmula pode ser simplificada para as equações (3), (2) ou (1), perdendo precisão em cada uma delas.

2.2. Segunda Etapa: Casos de uso expandidos

A etapa seguinte do processo de análise consiste na expansão dos casos de uso, ou seja, a listagem dos passos que compõe o fluxo principal e em seguida os fluxos alternativos. Considerando esta nova informação é possível adicionar mais detalhe à estimativa de esforço para produzir melhores resultados. Nesta fase da análise, portanto, será mostrado como aprimorar a estimativa dos valores $complex(uc)$, a partir da observação do caso de uso expandido. Porém, aqui será feita uma diferenciação entre os métodos de contagem usuais [6][7], pois não serão contabilizados todos os passos do caso de uso, mas somente aqueles que efetivamente geram complexidade [16].
De acordo com o desenvolvimento do método de Larman [8], pode-se observar que os diferentes passos descritos do caso de uso, darão origem às transações com o sistema. Cada transação de entrada, deve ser modelada e implementada como uma operação de sistema, que opera transformação nos dados. Já as transações de saída, devem ser modeladas como consultas, que apenas retornam informações já armazenadas no sistema de forma estruturada.
Outras transações poderão ter nenhuma influência sobre o tempo de desenvolvimento. Por exemplo, transações entre atores (que não são implementadas) ou transações que descrevem ações do sistema e que não implicam na troca de informações com o ambiente, como navegar de uma tela para outra ou exibir a confirmação da realização com sucesso de uma operação de sistema. Essas transações são consideradas por Wazlawick [17] como "passos complementares" e não são contabilizadas para efeito do cálculo da complexidade de um caso de uso. Apenas as transações de entrada e saída, associadas aos assim chamados "passos obrigatórios" são contabilizadas.
Como as transações de entrada e saída devem ser necessariamente implementadas, elas geram impacto na complexidade do caso de uso. Usando-se o mesmo raciocínio para definir o tempo de desenvolvimento de um sistema baseado no número de casos de uso, pode-se definir o tempo de desenvolvimento de um caso de uso baseado no número de transações obrigatórias:

$$(5)\ complex(uc) = f_{uc}\sum_{i=1}^{N_{uc}} complex_{uc}(to_i)$$

Onde: $complex_{uc}(to)$ é uma função que estabelece uma medida de complexidade para uma transação obrigatória, N_{uc} é o número de transações obrigatórias do caso de uso uc e f_{uc} é o fator de aumento esperado no número de transações obrigatórias no caso de uso nas fases posteriores da análise.
Aplicando-se a fórmula 5 em 4', obtém-se uma equação para estimação de esforço que considera não só o número de casos de uso, mas também a complexidade individual de cada caso de uso, calculada a partir do número de transações obrigatórias, sua complexidade estimada e o fator de aumento do número destas transações no caso de uso à medida que a análise prossegue:

$$(6)\ TDE = TFI + T * f\sum_{i=1}^{N}\left(f_{uc_i}\sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j)\right)$$

Se f_{uc} for definido como a média histórica, observada do aumento do número de transações obrigatórias dentro de cada caso de uso no momento atual da análise (etapa de expansão dos casos de uso) até a conclusão do sistema, então a equação 6 pode ser simplificada para:

(6') $TDE = TFI + T * f * f_{uc} \sum_{i=1}^{N} \sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j)$

O tempo estimado para desenvolvimento de um sistema será dado, então, pelo tempo necessário para desenvolver atividades independentes do número de casos de uso do sistema (TFI), somado ao tempo médio para desenvolver todos os artefatos derivados de um caso de uso (T), multiplicado pelo fator de aumento esperado no número de casos de uso ao longo do projeto (f), multiplicado pelo fator de aumento médio no número de passos obrigatórios dentro dos casos de uso (f_{uc}) multiplicado pela complexidade total de passos obrigatórios contabilizados em todos os casos de uso expandidos ($\sum_{i=1}^{N} \sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j)$). Seja N_{uc} o número total de passos obrigatórios nos casos de uso de um sistema, dado por $N_{uc} = \sum_{i=1}^{N} N_{uc_i}$.

Se todos os passos obrigatórios fossem igualmente complexos, e expressos por um valor médio α, então a equação 6' se reduziria a:

(6'') $TDE = TFI + T * f * f_{uc} * \alpha * N_{uc}$

Ou seja, a complexidade média de um caso de uso seria dada por uma função de 3 variáveis, que são ajustadas através de médias históricas (f, f_{uc}, e α) e o número total de passos obrigatórios nos casos de uso do sistema.

2.3. Terceira Etapa: Contratos

Mais detalhes serão acrescentados ao sistema quando forem definidos os contratos de cada operação e consulta de sistema (transações obrigatórias) [8]. Nesta fase, será possível detalhar mais a estimativa de complexidade individual da operação ou consulta de sistema $complex_{uc}(to)$.

Uma transação obrigatória corresponderá a uma operação ou consulta de sistema [17] com uma assinatura (onde são especificados os parâmetros). Cada operação ou consulta de sistema implica na existência de um ou mais contratos. Cada contrato possui pré-condições e pós-condições (apenas se for operação de sistema) ou resultados (apenas se for consulta de sistema). Todos estes elementos poderão afetar a complexidade da transação.

Os valores a serem considerados para cada transação são:

- n_{par} = número de parâmetros.
- n_{pre} = número de pré-condições.
- n_{pos} = número de pós-condições.
- n_{res} = número de resultados.

Cada um destes valores terá alguma influência na complexidade da transação de sistema. Porém o peso de cada valor poderá ser diferente. É necessário, portanto, não apenas somar os valores, mas obter uma soma ponderada com pesos que devem ser determinados experimentalmente:

(7) $complex_{uc}(to) = p1 * n_{par}(to) + p2 * n_{pre}(to) + p3 * n_{pos}(to) + p4 * n_{res}(to)$

Assim, na fase final da análise, quando os contratos já foram elaborados, a estimação de esforço total pode ser definida a partir da aplicação da equação (7) em (6'):

(8) $TDE = TFI + T * f * f_{uc} \sum_{i=1}^{N} \sum_{j=1}^{N_{uc_i}} \left(p1 * n_{par}(to_j) + p2 * n_{pre}(to_j) + p3 * n_{pos}(to_j) + n4 * n_{res}(to_j) \right)$

Esta equação consiste, portanto, em uma aproximação da complexidade de um sistema quando sua análise foi concluída. Inicia-se então a fase de projeto.

2.4 A Fase de Projeto e Implementação

Na fase de projeto, segundo Larman [8] e Wazlawick [17], cada transação obrigatória dará origem a um diagrama de comunicação, cuja complexidade dependerá do número de parâmetros, pré e pós-condições e resultados.

No caso das operações de sistema, a complexidade será muito mais dependente do número de pós-condições, pois a cada pós-condição corresponderá uma ação que precisa ser definida no diagrama.

A revisão do modelo conceitual e sua transformação em um diagrama de classes de projeto é um fator independente e pode ser contabilizada desta forma.

O projeto gráfico das interfaces terá forte dependência com o número de parâmetros e resultados das transações obrigatórias.

O projeto do banco de dados será um fator independente assim como a geração do diagrama de classes de projeto, visto que é possível automatizar esta atividade. Mas mesmo não sendo automatizada, ela não depende da complexidade interna das transações, embora possa estar relacionada com o número de casos de uso. O número de classes normalmente está diretamente relacionado ao número de casos de uso.
A fase de implementação pode ser altamente automatizada, caso se disponha de ferramentas adequadas. Este grau de automatização é um fator dependente, pois quanto menos automatizada for a geração de código, maior o tempo gasto com esta atividade e este tempo é proporcional à complexidade das operações de sistema, especialmente as pós-condições.

3. ESTUDO DE CASO

Foi realizado um estudo de caso para obter valores preliminares para as variáveis que o método necessita para ser aplicado na fase de concepção.
Os valores destas variáveis deverão variar conforme o projeto e a equipe, sendo importante, portanto, caracterizar o tipo de projeto e equipe.
Neste estudo de caso a equipe era experiente na linguagem de programação e no método de desenvolvimento (tendência para um valor baixo de TFI), mas tinha pouca experiência com a ferramenta CASE (tendência para um aumento no TFI). O sistema era bem compreendido, a análise era detalhada e um número significativo de casos de uso já estavam listados e expandidos no início do trabalho (tendência de que f seja próxima de 1). O projeto (um sistema de cadastro e emissão de relatórios pela *web*) foi desenvolvido até o final e o produto entregue ao cliente. Em função do tempo real de desenvolvimento deste projeto, os valores das variáveis foram calculados para esta equipe e este projeto.
O tempo total de desenvolvimento TD do sistema foi de 111 horas e cinco minutos. Na fase de concepção cinco casos de uso foram listados, e ao final do projeto mais um caso de uso foi descoberto. As variáveis da equação (4) foram assim calculadas:

TFI = 23 horas e 49 minutos (tempo gasto com atividades que não dependem da quantidade de casos de uso).

f = 6/5 = 1,2 (fator de aumento do número de casos de uso ao final do desenvolvimento em relação ao valor inicialmente previsto)

$complex(uc) = 1$ para todos os casos de uso.

T = (111h05 – 23h49) / 1,2*5 = 12h07m13 (calculado como $(TD - TFI) / f * N$)

Portanto, nesta etapa calculou-se que o tempo médio para o desenvolvimento dos artefatos derivados de um caso de uso deste tipo de sistema com esta equipe foi de aproximadamente 12 horas e 07 minutos.
Com estes dados apenas a equação de estimação de esforço da fase de concepção poderia ser definida como:

(9) $TDE = TFI + 12{,}7 * 1{,}2 * N = TFI + 15{,}24 * N$

Portanto, para cada caso de uso identificado por esta equipe em futuros sistemas do mesmo tipo, 15,24 horas em média serão necessárias para desenvolvimento. O valor de TFI não depende do número de casos de uso do sistema e deve, portanto, ser estimado separadamente. Pressupõe-se que, com o aumento da experiência da equipe esse valor 15,24 vá diminuindo até atingir um mínimo, que corresponde à máxima performance possível da equipe com a tecnologia de desenvolvimento disponível. Investimentos em treinamento, melhoria de processo e automatização aumentarão inicialmente o valor de TFI, mas com o tempo diminuirão o limite mínimo de $T * f$, correspondendo, portanto a ganhos permanentes.

4. TRABALHOS RELACIONADOS

Estimativas são exigidas quando é preciso planejar e gerenciar projetos de software [10]. Em muitos casos, as estimativas são feitas a partir de bases históricas e experiências de especialistas, mas também existem modelos de estimação formais para serem utilizados. Exemplos desses modelos são: SLOC (Source Lines of Code) [10], FP (Function Points) [15] e UCP (Use Case Points) [6][7].
O método UCP foi proposto principalmente por causa da dificuldade de avaliar SLOC e FP para o desenvolvimento de software baseado no PU (Processo Unificado) e UML (Unified Modeling Language).
O UCP [6][7] foi proposto por Karner em 1993, a fim de gerar estimativa de esforço para software baseado em casos de uso.
A contagem de Pontos de Caso de Uso considera alguns índices:

- Primeiramente, os atores são categorizados em simples, médios e complexos. O Peso não Ajustado de Atores (UAW) corresponde à soma dos pesos de todos os atores.
- Os casos de uso também são caracterizados como simples, médios e complexos, dependendo da quantidade de passos. O Peso não Ajustado de Casos de Uso (UUCW) é calculado como sendo a soma dos pesos de todos os casos de uso. Então o UAW é adicionado ao UUCW para produzir os Pontos de Caso de Uso não Ajustados (UUCP).

- Os Pontos são então ajustados de acordo com os Fatores de Complexidade Técnica (TCF) e fatores ambientais (EF). Fatores ambientais e seus pesos foram importados da teoria de Pontos de Função, e os fatores de complexidade técnica foram propostos por [6][7] após entrevistas com analistas. Assim, UCP = UUCP * TCF * EF.

Tabela 1: Fatores de Complexidade Técnica

Descrição	Peso
Sistemas Distribuídos	2.0
Desempenho da aplicação	1.0
Eficiência do usuário final (on-line)	1.0
Processamento interno complexo	1.0
Reusabilidade do código em outras aplicações	1.0
Facilidade de instalação	0.5
Usabilidade (facilidade operacional)	0.5
Portabilidade	2.0
Facilidade de manutenção	1.0
Concorrência	1.0
Características especiais de segurança	1.0
Acesso direto para terceiros	1.0
Facilidades especiais de treinamento	1.0

Tabela 2: Fatores Ambientais

#	Descrição	Peso
EF1	Familiaridade com o Processo de Desenvolvimento de Software	1.5
EF2	Experiência na Aplicação	0.5
EF3	Experiência com OO, na Linguagem e na Técnica de Desenvolvimento	1.0
EF4	Capacidade do Líder de Projeto	0.5
EF5	Motivação	1.0
EF6	Requisitos estáveis	2.0
EF7	Trabalhadores com dedicação parcial	-1.0
EF8	Dificuldade da Linguagem de Programação	-1.0

Os Fatores de Complexidade Técnica foram descartados há poucos anos atrás porque foi decidido que eles não fazem sentido no desenvolvimento de software atual [11] [12] [15].
Os Fatores Ambientais estão relacionados com a equação 4' da seguinte forma:

- EF1: Quanto maior a familiaridade com o processo de desenvolvimento (EF1), menor o tempo TFI e menores os valores de f e $complex$.
- EF2: Experiência com a aplicação (EF2) implica em um menor valor para f, pois como os casos de uso já serão conhecidos existe uma possibilidade mais baixa de que os analistas esqueçam de alguns deles.
- EF3: Experiência com OO na Linguagem e na Técnica de Desenvolvimento (EF3) terá impacto em $complex$, especialmente na fase de geração e otimização de código.
- EF4: Quanto maior capacidade do líder do projeto (EF4) menores os valores esperados para f e $complex$. Mas se a capacidade da equipe for baixa, isso implicará em um aumento no tempo TFI, pois o líder perceberá a necessidade de treinamento e investirá nisso.

- EF5: A motivação da equipe (EF5) em si não tem influência sobre as variáveis, mas pode intensificar a influência de outros aspectos. Por exemplo, uma equipe motivada e familiarizada com o processo tenderá a determinar um valor de *complex* menor do que uma equipe familiarizada com o processo mas não motivada.
- EF6: Requisitos estáveis (EF6) terão influência mais significativa sobre f. Mas o gerenciamento dos riscos associados à mudança dos requisitos pode fazer com que *TFI* cresça na medida que os requisitos forem menos estáveis.
- EF7: Trabalhadores com dedicação parcial (EF7) não afetam diretamente as variáveis. Dependendo das atividades que estes trabalhadores desempenham em paralelo, o impacto poderá ser positivo ou negativo no processo como um todo.
- EF8: Uma linguagem de programação difícil de usar ou inadequada (EF8) certamente terá impacto crescente em *complex*.

Assim, os fatores da equação 4' ainda poderiam ser refinados considerando-se os fatores ambientais de UCP. A diferença entre os dois métodos consiste em que UCP apenas multiplica os valores dos pesos destes fatores pela complexidade estimada dos casos de uso, enquanto que o método proposto apresenta um ajuste muito mais fino da influência destes fatores na estimativa de esforço.

5. CONCLUSÕES

A teoria apresentada aqui considera a estrutura do processo de desenvolvimento orientado a objetos como determinante para a estimativa de esforço em várias fases do processo de análise, diferente do método UCP, que apenas aplica a multiplicação dos fatores ambientais sobre a complexidade estimada dos casos de uso e atores.

A equação 8 pode ser mais refinada, com o desdobramento da função interna aos somatórios em valores associados aos tempos de projeto, geração de código e testes. Os diferentes valores dos pesos associados a essa função podem ser obtidos, a partir de uma massa de dados suficientemente grande, pela correlação observada, mas isso é uma proposta para trabalhos futuros.

Neste momento é possível afirmar que os diferentes fatores ambientais do método Pontos de Caso de Uso não devem ser tratados apenas como índices (daí a baixa precisão usualmente associada a este método), igualmente ao método de Pontos de Função [1], mas como fatores que determinam aumentos ou decréscimos nas diferentes variáveis da equação de estimativa.

Neste artigo esses índices foram calculados com base na equação 4', ou seja, no contexto da fase de concepção, mas é possível estender ainda mais esta análise se for usada a equação 6'', da fase de expansão dos casos de uso, ou a equação 8, aplicável depois que os contratos já estão escritos.

Agradecimentos

Esta pesquisa foi realizada com uma bolsa de estudo da agência CAPES.

Referências

[1] Albrecht, A. J. Measuring Application Development Productivity. Proceedings of the IBM SHARE/GUIDE Applications Development Symposium.Monterey, CA; (October, 1979). (pp. 83-92).

[2] Anda, B. Comparing effort estimates based on use cases with expert estimates. In Proceedings of Empirical Assessment in Software Engineering (EASE 2002) (Keele, UK, (April, 2002), 8-10), 13 p.

[3] Anda, B., Dreiem, D., Sjøberg, D.I.K., Jørgensen, M. Estimating software development effort based on use cases - Experiences from industry. In M. Gogolla, C. Kobryn (Eds.): UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, 4th Int'l Conference. Springer-Verlag LNCS 2185, (2001), 487-502.

[4] Anda, B, Sjøberg, D.I.K. and Jørgensen, M. Quality and Understandability in Use Case Models, In J. Lindskov Knudsen (Ed.): ECOOP 2001 - Object-Oriented Programming, 15th European Conference, Budapest, Hungary, (June 18-22, 2001), LNCS 2072 Springer-Verlag, pp. 402–428.

[5] Cockburn, A. *Escrevendo Casos de Uso Eficazes: um guia prático para desenvolvedores de software*. Trad. Roberto Vedoato. Porto Alegre: Bookman. 2005.

[6] Karner, G. Resource Estimation for Objectory Projects. Objectory Systems. September, 1993.

[7] Karner, G. Metrics for Objectory. thesis at the university of Linkopmg, Sweden. LiTH-IDA-Ex-9344; 21. December, 1993.

[8] Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. Prentice-Hall, second edition, 2002.*

[9] Longstreet, D. Use cases and function points. Copyright Longstreet Consulting Inc. 2001. www.softwaremetrics.com.

[10] Pressman, Roger S. *Engenharia de Software*. São Paulo: Makron Books. 1995.

[11] Ribu, K. Estimating object oriented software projects with use cases. Oslo: University of Oslo, 2001, 132 p. Master of Science - Thesis - Department of Informatics.

[12] Rule, P.G. Using measures to understand requirements. Software Measurement Services Ltd. 2001.

[13] Smith, J. The estimation of effort based on use cases. Rational Software White Paper. 1999.

[14] Symons, C.R. Software Sizing and Estimating, MKII FPA. John Wiley and Sons. 1991.

[15] Vazquez, C; Simões, G.S e Albert, R.M. *Análise de Pontos de Função: medição, estimativas e gerenciamento de projetos de software*. 1. ed. São Paulo: Érica. 2003.

[16] Vieira, E.L.; Wazlawick, R.S. Improving the Use Case Points Method by Counting only Mandatory Steps. Technical Report: LSC-UFSC. 2005.

[17] Wazlawick, R. S. Análise e Projeto de Sistemas de Informação Orientado a Objetos. Rio de Janeiro: Elsevier. 2004.