Uma Metodologia para Estimativa de Esforço em Projetos de Softwares Desenvolvidos com ICONIX Empregando *Use Case Points*

Fabrício S. Melo[1]

[1]Universidade Estácio de Sergipe (FASE)
Rua Teixeira de Freitas, 10, Salgado Filho – Aracaju, SE - Brasil
fs-melo@bol.com.br

Abstract. *This paper presents a methodology for estimating the development effort of object-oriented software, still in the phase of requirements analysis, using Use Case Points (UCP) applied to projects executed with the ICONIX process. A procedure as an alternative to arbitrary estimates typically found in agile software development was presented. Although the technique of UCP is not as widespread as the function point analysis, this study showed that it is still quite useful to be easily achievable in the first phase of the process under study, ensuring thereby a systematic estimation, adaptive to changes requirements and extremely rapid, contributing to software quality without neglecting the agile philosophy.*

Resumo. *Este artigo apresenta uma metodologia para estimar o esforço no desenvolvimento de software orientado a objetos, ainda na fase de análise de requisitos, utilizando Use Case Points (UCP) aplicada a projetos executados com o processo ICONIX. Foi apresentado um procedimento em alternativa às estimativas arbitrárias normalmente encontradas em desenvolvimento ágil de software. Apesar da técnica de UCP não ser tão difundida quanto à análise de pontos de função, esse estudo mostrou que ela ainda é bastante útil por ser facilmente realizável na primeira fase do processo em estudo, garantindo, com isso, uma estimativa sistematizada, adaptável às mudanças de requisitos e extremamente rápida, contribuindo para qualidade do software sem deixar de lado a filosofia ágil.*

Introdução

Nos últimos anos o desenvolvimento ágil de software foi proposto como solução para muitos dos problemas enfrentados no trabalho em projetos de software. Essa filosofia preconiza o desenvolvimento iterativo e incremental de softwares que evoluem através da colaboração de equipes multifuncionais motivadas e auto-organizadas que têm como missão promover um planejamento adaptativo e respostas rápidas e flexíveis às mudanças, principalmente de requisitos, ao longo do ciclo de vida do software. Sob essa ótica o ICONIX define-se como um processo de desenvolvimento de software ágil situado entre a alta complexidade, formalismo e abrangência do RUP (*Rational Unified Process*) e a simplicidade do XP (*Extreme programing*) [Rosenberg & Stephens 2007; Rosenberg, Stephens & Collins-Cope 2005].

Assim como o RUP, o ICONIX é um processo dirigido por casos de uso UML (*Unified Modeling Language*) que guiam o esforço de desenvolvimento com alta rastreabilidade de requisitos, no entanto mais leve que este. É tão leve e simples quanto o XP, mas não dispensa as tarefas de análise e projeto. Utiliza apenas quatro diagramas

UML. Por essa razão, o processo ICONIX é bem adequado para projetos ágeis, onde o *feedback* rápido é necessário, tais como atividades de requisitos, projeto, e estimativas.

Segundo Pressman (2006) ao início de um projeto de software são necessárias elaborações de um plano organizado e de estimativas quantitativas, mas nesse momento geralmente não há informação sólida disponível. Para o autor uma análise detalhada dos requisitos de software pode fornecer a informação necessária para as estimativas, mas a análise frequentemente prolonga-se por longos períodos na execução do projeto. Quanto ao plano de desenvolvimento, esse pode ser elaborado examinando o produto e o problema que ele pretende resolver no início do projeto para então proceder a elaboração de seu escopo. A qual é a primeira atividade de gestão de um projeto de software.

Em resposta ao problema da elaboração do escopo no início do desenvolvimento e do elevado tempo para tanto, alguns processos ágeis, como ICONIX, podem remediar o embate. No entanto, para além da determinação de um escopo, o bom e eficiente planejamento deverá apresentar estimativas fidedignas no início da execução de um projeto para que as atividades de monitoramento e controle possam prosseguir de forma a garantir um produto final que atende os prazos e custos acordados. Nesse sentido o presente trabalho objetiva estabelecer uma metodologia para estimar o esforço no desenvolvimento de software orientado a objetos, ainda na fase de análise de requisitos, utilizando *Use Case Points* (UCP).

Visão Geral do Processo ICONIX

Segundo Rosenberg e Stephens (2007), mentores do processo, seu principal objetivo é estabelecer uma sequência de passos para transformar casos de uso em código fonte limpo de forma ágil. Para tanto os fluxos de trabalhos são divididos em dinâmicos e estáticos, e podem passar de uma iteração de todo o processo para um pequeno grupo de casos de uso. Do ponto de vista temporal, os autores dividem o ciclo de desenvolvimento em quatro fases e apontam para as atividades que devem executadas em cada uma. São elas:

Requisitos

Nessa fase são identificados os casos de uso envolvidos, criada uma lista de requisitos funcionais e elaborado o modelo de domínio composto por um diagrama de classes de alto nível que em seguida são associadas aos requisitos funcionais e aos casos de uso.

Análise e projeto preliminar

Aqui é desenhado um diagrama de robustez, reescrito cada caso de uso, e atualizado o modelo de domínio com atributos das classes e novos objetos encontrados.

Projeto detalhado

Agora são desenhados diagramas de sequência (um por caso de uso) para mostrar em detalhes o que vai ser implementado, atualizado o modelo de domínio e adicionadas operações para os objetos de domínio.

Implementação

Por fim é escrito o código e os testes de unidade, ou, a critério, escreve-se os testes de unidade e, em seguida, o código; seguindo com os testes de integração e uma revisão de código e modelo de atualização para se preparar para a próxima iteração de desenvolvimento.

Contudo, conforme inclusive sintetiza Silva e Videira (2001), o ICONIX consiste na produção de artefatos que retratam as visões dinâmicas e estáticas de um sistema e que esses modelos são desenvolvidos em paralelo durante todo ciclo de vida do processo, de acordo com a Figura 1.

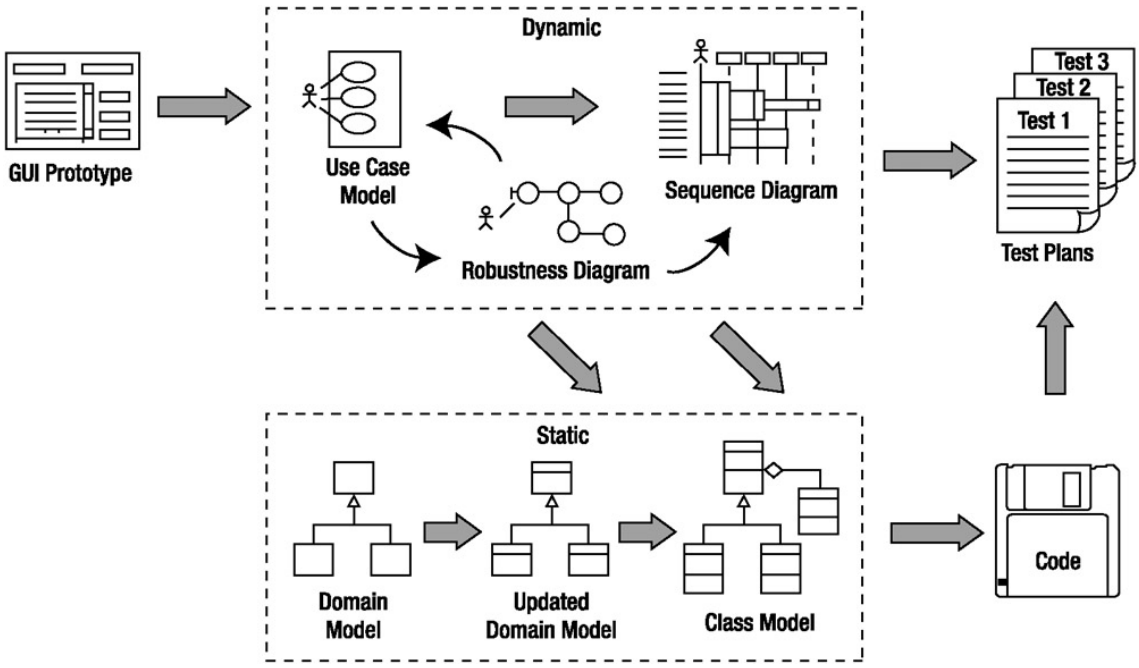

Figura 4 - Visão geral do ICONIX [Rosenberg e Stephens 2007]

Determinação do Escopo

Modelos de casos de uso são usados em análise orientada a objetos para capturar e descrever os requisitos funcionais de um sistema. Considerados artefatos distintos, mas intimamente ligados, um requisito funcional e um caso de uso podem associar-se em uma relação de muitos para muitos (*n:n*). Essa colocação é muito importante na determinação do escopo do sistema que será projetado, pois ao final da fase de requisitos uma lista que os associa à(s) classe(s) de domínio servirá para orientar o processo de desenvolvimento demonstrando o que deve ser produzido e, de forma bastante dispersa, alguma informação sobre o esforço a ser despendido na execução do projeto, já que cada caso de uso estará associado à produção ou até mesmo a reutilização, de um número de classes, conforme exemplo na Tabela 1.

Tabela 1 - Lista de associação entre requisitos, casos de uso e classes

Requisitos	Casos de Uso	Classes a Produzir	Classes Reutilizadas
R1, R2, R3	CU-01	CL-01, CL-02	
R2, R4, R5	CU-02	CL-03	CL-01
R1, R2, R6, R7	CU-03	CL-04, CL-05	CL-01

No entanto, apesar de demonstrar um escopo candidato ao projeto, a lista acima ainda não apresenta uma forma sistemática para inferir estimativas iniciais que seriam usadas para planejar prazos e custos. O que pode ser solucionado com utilização da contagem de *Use Case Points* (UCP). Um método de estimativa de esforço proposto por Karner (1993), o qual se inspirou no modelo de Pontos de Função proposto por Albrech (1979) e posteriormente melhorado por Symons (1988).

Estimativa de Esforço em Projetos de Softwares com *Use Case Points*

O modelo de contagem por casos de uso proposto por Karner, o UCP, traz o benefício de poder ser executado ainda na fase de requisitos, mas foi criado para aplicação no processo *Objectory*. Ele começa com a medição da funcionalidade do sistema baseado no modelo de casos de uso em uma contagem não ajustada chamada *Unadjusted Use Case Point* (UUCP). Nesse momento são avaliados fatores de complexidade técnica (TCF) envolvidos no desenvolvimento desta funcionalidade. O último passo na estimativa é avaliar o fator ambiental (EF) proposto pelo autor. O número de UCP é dado então pela Equação 1 e deverá ser multiplicado pela taxa média de esforço para encontrar a estimativa de esforço total do projeto.

$$UCP = UUCP \times TCF \times EF \qquad \text{(Equação 1)}$$

Em seu estudo Karner (1993) analisou três projetos estimados com UCP, os quais apresentaram uma taxa média de esforço entre 20,0 e 28,7 homem-hora/UCP. Apesar de muito pequena a mostra, notou-se a simplicidade, versatilidade e capacidade de adaptabilidade dessa técnica. Mais tarde as experiências de Schneider e Winters (2001) corroboraram com isso encontrando uma taxa de 20 ou 28 homem-hora/UCP segundo a simplicidade ou complexidade do sistema, respectivamente.

Segundo Kress *et al* (2014) o fato do método de Pontos de Caso de Uso nunca ter sido completamente calibrado usando análise de regressão, devido à falta de um número estatisticamente suficiente de projetos, torna-o passível de uma grande fraqueza. No entanto, o dimensionamento de fácil aplicação e regras de contagem proporciona muitos benefícios para estimativas em fases iniciais e, assim, permite medir rapidamente o tamanho funcional de uma aplicação.

Contagem de *Unadjusted Use Case Points* (UUCP)

O UUCP é calculado para aferir a cada ator e cada caso de uso um grau de complexidade, simples, médio ou complexo, através da aplicação das regras da Tabela 2 e da Tabela 3, respectivamente.

Tabela 2 - Ponderação dos atores, adaptada de Karner, 1993

Complexidade	Definição	Peso(W_i)
SIMPLES	Um ator é simples, se ele representa outro sistema com uma interface de programação de aplicativo definido.	1
MÉDIO	Um ator é médio se: 1. Uma interação com outro sistema através de um protocolo. 2. Uma interação humana com um terminal de linha.	2
COMPLEXO	Um ator é complexo se ele interage através de uma interface gráfica do usuário.	3

Tabela 3 - Ponderação dos casos de uso, adaptada de Karner, 1993

Complexidade	Definição	Peso(W_i)
SIMPLES	Um caso de uso é simples, se tiver 3 ou menos transações, incluindo cursos alternativos. Você deve ser capaz de perceber o caso de uso com menos de 5 objetos de análise.	5
MÉDIO	Um caso de uso é médio se tem de 3 a 7 transações, incluindo cursos alternativos. Você deve ser capaz de perceber o caso de uso com 5 a 10 objetos de análise.	10
COMPLEXO	Um caso de uso é complexo se ele tem mais de 7 transações, incluindo cursos alternativos. O caso de uso deve, no mínimo, possuir 10 objetos de análise a serem realizados.	15

Agora se devem somar os pesos dos atores e dos casos de uso em conjunto para obter o UUCP, conforme Equação 2.

(Equação 2)

$$UUCP = \sum_{i=1}^{6} n_i \cdot W_i$$

Onde n_i é o número de itens de variedade *i*.

Contagem do Fator de Complexidade Técnica (TCF)

O fator de complexidade técnica (TCF) varia de acordo com a dificuldade de construção do sistema e é calculado pela Equação 3.

(Equação 3)

$$TCF = C_1 + C_2 \sum_{i=1}^{6} F_i \cdot W_i$$

Onde $C_1 = 0{,}6$, $C_2 = 0{,}01$ e F_i é um fator que é classificado numa escala de 0, 1, 2, 3, 4 e 5, conforme Tabela 4. O peso 0 significa que o fator não é relevante e 5 significa que é essencial.

Tabela 4 - Fatores de Complexidade Técnica, adaptada de Karner, 1993

F_i	Fatores de Contribuição na Complexidade	Peso(Wi)
F_1	Sistemas distribuídos.	2
F_2	Objetivos de desempenho do aplicativo, em qualquer resposta ou transferência.	1
F_3	Eficiência do usuário final (*on-line*).	1
F_4	Complexidade de processamento interno.	1
F_5	Reutilização, o código deve ser capaz de reutilizar em outras aplicações.	1
F_6	Facilidade de instalação.	0.5
F_7	Facilidade operacional, facilidade de utilização.	0.5
F_8	Portabilidade.	2
F_9	Mutabilidade.	1
F_{10}	Concorrência	1
F_{11}	Recursos de segurança especiais.	1
F_{12}	Proporcionar o acesso direto para terceiros.	1
F_{13}	Facilidades de treinamento do usuário especial	1

Contagem do Fator ambiental (EF)

O fator ambiental (EF) nos ajuda a estimar o quão eficiente é o nosso projeto. E é calculado pela Equação 4. (Equação 4)

$$EF = C_1 + C_2 \sum_{i=1}^{8} F_i \cdot W_i$$

Onde C_1 = 1.4, C_2 = -0.03 e F_i , assim como no TCF, é um fator que é classificado numa escala de 0, 1, 2, 3, 4 e 5, conforme Tabela 5. O peso 0 significa que o fator não é relevante e 5 significa que é essencial.

Tabela 5 - Fatores ambientais, adaptada de Karner, 1993

F_i	**Fatores Contribuição na Eficiência**	**Peso(W_i)**
F_1	Familiarizado com o processo	1.5
F_2	Trabalhadores a tempo parcial	-1
F_3	Capacidade do Analista	0.5
F_4	Experiência de aplicação	0.5
F_5	Experiência em orientação a objetos	1
F_6	Motivação	1
F_7	Dificuldade na linguagem de programação	-1
F_8	Requisitos estáveis	2

Resultados e Discussão

A metodologia proposta para estimativa de projetos desenvolvidos com ICONIX através da contagem de UCP é apresentada pelo procedimento abaixo:

1. Produzir uma tabela listando as associações entre requisitos funcionais, casos de uso e classes de domínio, conforme modelo da Tabela 1;
2. Calcular os pontos de pontos de casos de uso não ajustados (UUCP), ponderando a Tabela 3 com base no número de classes de domínio detectado na Tabela 1, excluindo-se aquelas reutilizadas, já implementadas anteriormente;
3. Calcular o fator de complexidade técnica (TCF);
4. Calcular o fator ambiental (EF);
5. Calcular os pontos de casos de uso (UCP);
6. Calcular o esforço médio (EM) multiplicando os UCPs calculados pela taxa média de esforço (homen-hora/UCP) (TME) observada nas estatísticas de projetos anteriores para encontrar a quantidade de recursos necessários (em homem-hora) conforme Equação 5.

$$EM = UCP \times TEM \quad \text{(Equação 5)}$$

Para ilustrar o procedimento de cálculo da UCP, um sistema de pedidos *on-line* será utilizado. A Figura 2 mostra o diagrama de caso de uso para o sistema em desenvolvimento, esse deverá interagir através de uma interface gráfica com três atores, os quais são classificados como complexos.

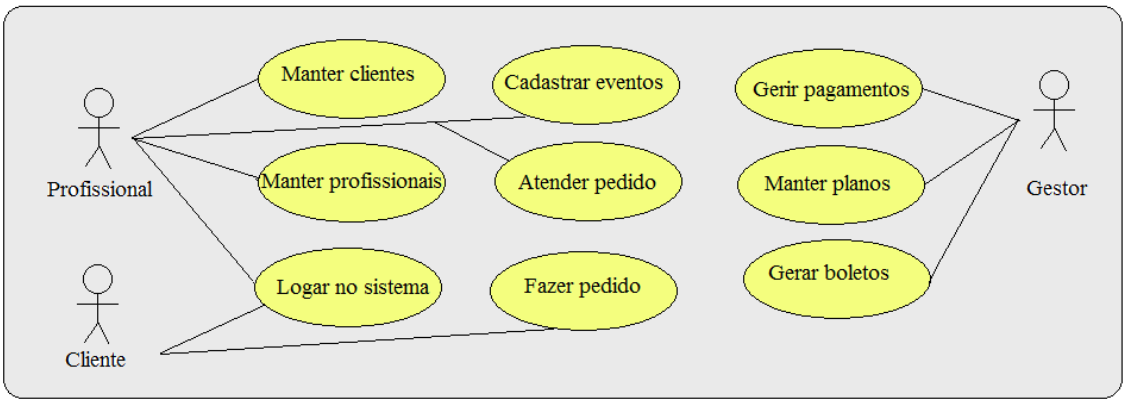

Figura 5 - Sistema de Pedidos *on-line*

Na construção da tabela de associações, seguindo a Tabela 1 de exemplo, notou-se que em todos os casos de uso panejados não se observa mais de dois ou três objetos de domínio. Sendo assim os nove casos de uso apresentados são classificados como simples. Dessa forma os pontos de pontos de casos de uso não ajustados foram calculados pela Equação 2 da seguinte forma $UUCP = (3 \times 3) + (9 \times 5) = 54$.

Calculando o fator de complexidade técnica através da Equação 3 temos $TCF = 0{,}6 + 0{,}01 \times \{(2 \times 0) + (1 \times 3) + (1 \times 3) + (1 \times 2) + (1 \times 3) + (0{,}5 \times 0) + (0{,}5 \times 3) + (2 \times 0) + (1 \times 0) + (1 \times 3) + (1 \times 3) + (1 \times 0) + (1 \times 0)\} = 0{,}785$

Calculando o fator ambiental através da Equação 4 temos $EF = 1{,}4 - 0{,}03\{(1{,}5 \times 0) + (-1 \times 0) + (0{,}5 \times 4) + (0{,}5 \times 0) + (1 \times 4) + (1 \times 4) + (-1 \times 0) + (2 \times 5)\} = 0{,}8$
.

Utilizando a Equação 1, fazendo $UCP = UUCP \times TCF \times EF$, temos $UCP = 54 \times 0{,}785 \times 0{,}8 = 33{,}9$.

Como pode ser visto abaixo foi encontrado o esforço médio (EM) pela aplicação da Equação 5 usando uma taxa média de esforço de 24 homen-hora/UCP, conforme encontrado na literatura [Schneider e Winters 2001].

$$EM = UCP \times TEM = 33{,}9 \times 24 = 813{,}6 \text{ homen-hora}$$

Por fim, o processo de estimativas em discussão deverá ser melhorado a cada final de iteração do processo ICONIX adaptando a taxa média de esforço ao histórico de projetos da equipe de desenvolvimento, no entanto, de início poderão ser usadas taxas como descritas na literatura citada anteriormente, entre 20 e 28 homem-hora/UCP.

Conclusão

A metodologia apresentada trouxe uma alternativa às estimativas arbitrárias normalmente encontradas em desenvolvimento ágil de software. Apesar da técnica de UCP não ser tão difundida quanto à análise de pontos de função, esse estudo mostrou que ela ainda é bastante útil por ser facilmente realizável ainda na primeira fase do

processo em estudo, garantindo, com isso, uma estimativa sistematizada, adaptável às mudanças de requisitos e extremamente rápida, contribuindo para qualidade do software sem deixar de lado a filosofia "ágil".

Nota-se, então que a aplicação da técnica em questão (UCP) satisfaz os objetivos iniciais desse trabalho, e que novas pesquisas deverão explorar testes em maiores espaços amostrais, inclusive com o uso de práticas de inteligência artificial, primando pela contagem e ajuste dos pontos de casos de uso e pela validação estatística do modelo.

Referências

ALBRECHT, A. J (1979). Measuring application development productivity. Proc. of IBM Applic. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA, pp. 83-92.

KARNER, G. (1993), Resource Estimation for Objectory Projects – Objective Systems.

KRESS, C.F.; HUMMEL, O.; HUQ, M. (2014), A Practical Approach for Reliable Pre-Project Effort Estimation. *In*: CEUR Workshop Proceedings, Vol. 1138, p. 23.

PRESSMAN , R.S. (2011), Engenharia de Software, Rio de Janeiro: McGraw Hill, 6 ed.

ROSENBERG, D. & STEPHENS, M. (2007), Use Case Driven Object Modeling with UML: Theory and Practice. Apress.

ROSENBERG, D., STEPHENS, M. & COLLINS-COPE, M. (2005), Agile Development with ICONIX Process. Apress.

SCHNEIDER, G.,WINTERS, J.P. (2001), Applied Use Cases, A Practical Guide, Addison-Wesley, Reading, MA.

SYMONS, C. R. (1988), Function Point Analysis : Difficulties and improvements. *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 1. Jan.