# Melhoria de um Processo de Estimativa de desenvolvimento de software através do emprego de metodologia e criação da técnica de estimativa – API Points

**Ana Carina M. Almeida[1], Carla Renata R. Luz [1]**

[1]Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R)
Rua Bione, 220 - Bairro do Recife – CEP 50030-390
Recife - PE - Brasil

{ana.almeida, carla.ribeiro}@cesar.org.br

***Abstract.*** *By choosing a good estimation method is possible to control the projects resources with greater certainty, but due to the complexion of the software nature it is not possible to use all technical solution in all situations. This article has the purpose to present a new method of estimating effort of API (Application Program Interface), known as API Points. Besides that, this article explains the methodology used to creation this technique. API Points aims to estimate accurately and quickly, improving the planning of a development software project.*

***Resumo*** *Através da escolha de um bom método de estimativa é possível controlar os recursos de projetos com maior segurança, porém devido à natureza do software nem toda técnica se adapta a toda situação. Este artigo tem como propósito apresentar um novo método de estimativa de esforço de API (Application Program Interface), denominado API Points. Além de relatar a metodologia utilizada para criação desta técnica. Com a utilização deste método objetiva-se estimar de forma mais acurada e com agilidade, melhorando o planejamento de um projeto de desenvolvimento de software.*

## 1. Introdução

De acordo com o PMBOK [PMBOK, 2000] as etapas do processo de gerenciamento de projeto são: iniciação, planejamento, execução, monitoramento e controle, e encerramento. A etapa de planejamento contém uma série de atividades necessárias para dar início ao projeto, entre elas, pode-se mencionar estimativas de tempo, custo e esforço do projeto [Mulcahy, 2007].

A realização de uma estimativa permite que o gerente de projeto possa ajustar os parâmetros do projeto de acordo com o orçamento e prazos estipulados [Chen, 2005]. Caso algum parâmetro não possa ser ajustado, com a ajuda da estimativa é possível melhor gerenciar riscos, ao invés de ignorá-los ou ser surpreendido por eles posteriormente [Laird, 2006] [Boehm & Fairley, 2000].

Além disso, constata-se que com estimativas bem acuradas pode-se desenvolver um cronograma de atividades bem elaborado, efetivo e controlado, permitindo que as atividades de desenvolvimento do produto ocorram de forma equilibrada [McConnell, 2006].

Este benefício é conseguido pela utilização de um processo sistematizado de estimativa e técnicas adaptadas para a realidade do produto a ser desenvolvido. Sabe-se que não existe uma única técnica de estimativa que atenda a todas as necessidades devido à natureza dinâmica do software. Não existe um único modelo que proporcione uma estimativa acurada para todos os domínios [Laird & Brennan, 2006] [McConnell, 2006].

Este artigo tem como objetivo descrever a metodologia utilizada para criação de uma técnica de estimativa que se adéqua ao domínio, arquitetura e tecnologia de um Kit de Desenvolvimento de Software (em inglês, *Software Development Kit* - SDK), e também descrever a técnica desenvolvida, denominada API Points.

Este trabalho está organizado da seguinte forma: a seção 2 explica a necessidade de desenvolvimento de uma técnica de estimativa que atenda os requisitos de um Software Development Kit. A seção 3 descreve a metodologia usada para criação da nova técnica. A seção 4 descreve a técnica de estimativa, API Points. A seção 5 mostra uma conclusão sobre a pesquisa e relata trabalho futuro.

## 2. Motivação

O *Software Development Kit* (SDK), como o próprio nome diz, é um conjunto de ferramentas para desenvolvimento e teste de aplicações. O SDK tem como objetivo disponibilizar todas as funcionalidades necessárias para a construção de aplicativos (programas). O case deste artigo foca em um SDK para desenvolvimento de MIDlets, um aplicativo em linguagem de programação Java [Java, 2008] para dispositivos móveis. Esta ferramenta possibilita uma reprodução semelhante do comportamento de um dispositivo móvel real, incluindo seus recursos computacionais, tais como, uso de memória volátil, display e armazenamento de dados (mensagens e agenda de contatos, por exemplo); além de simular também a infra-estrutura de rede necessária para desenvolvimento de uma aplicação desta categoria.

O gerenciamento e planejamento do desenvolvimento de um aplicativo deste porte não é uma tarefa trivial e pequenos desvios na estimativa de esforço podem gerar impactos significativos na data de entrega do produto.

Para a realização de estimativas desse projeto foi utilizado o método Wideband Delphi. Este método conta com mais de um especialista no domínio da aplicação para realizar a estimativa e dar opinião sobre o esforço para realização do trabalho. O método facilita a convergência de opiniões entre os especialistas e desta forma acredita-se chegar a uma resposta mais acurada sobre o esforço [Laird & Brennan, 2006] [Mehwish, 2006].

Utilizando Wideband Delphi é possível chegar a uma estimativa com margem de desvio pequena desde que sejam envolvidos especialistas no assunto. [Laird & Brennan, 2006]. Por se tratar de um projeto de tecnologia e de domínio muito específico torna-se difícil a obtenção de um *expert* no assunto ou de uma base de dados histórica de um projeto maduro que possa servir de referência para o embasamento de uma nova estimativa.

Estes dois pontos fazem com que a utilização do Wideband Delphi em projetos desta categoria não traga maior segurança no planejamento e consequentemente na alocação de recursos.

A Tabela 1 exemplifica o esforço estimado e o esforço realizado de alguns componentes (API - *Application Program Interface*) do SDK. Na coluna "Erro Relativo" é constatado a variação entre o planejado *versus* realizado. Vale a pena mencionar que o Wideband Delphi foi utilizado durante o período de pré-venda (proposta) com o intuito de fornecer uma estimativa por ordem de grandeza. Para estimativas por ordem de grandeza é esperado constatar variações de -25% e 75% com relação ao que foi previamente estimado [Mulcahy, 2007].

**Tabela 1. Esforço Planejado x Esforço Realizado**

| API | Esforço Estimado | Esforço Realizado | Erro Relativo |
|---|---|---|---|
| API 01 | 704,00 | 64,00 | 10,00 |
| API 02 | 704,00 | 112,00 | 5,29 |
| API 03 | 352,00 | 260,00 | 0,35 |
| API 04 | 1056,00 | 463,00 | 1,28 |
| API 05 | 704,00 | 127,00 | 4,54 |

Através de uma estimativa por ordem de grandeza se ganha em agilidade no tempo de resposta, mas se perde em acurácia. Na Tabela 1, é possível visualizar o Erro Relativo (ER), calculado da seguinte forma: ER = (esforço estimado – esforço realizado)/esforço realizado [Chen, 2005], da estimativa das APIs. Todos os exemplos ultrapassam o limite de valores máximo (0,75) e mínimo (-0,25) indicados para estimativa de ordem de grandeza [Mulcahy, 2007]. O maior Erro Relativo foi da API 01, cerca de 10,00. Fato que acarretou numa alocação de recursos muito maior do que era realmente necessário para construção da API.

Outra desvantagem constatada do método Wideband Delphi é que não se trabalha com alguns dados quantitativos, como por exemplo, tamanho. Fato que torna mais difícil justificar a necessidade de se ter dez engenheiros trabalhando para entregar um produto durante um período de um ano [Laird & Brennan, 2006]. Além disso, faz-se necessário trabalhar com variáveis que possam indicar o desempenho do time, como por

exemplo, a produtividade do grupo ou de equipes. Observa-se que sem a medida de tamanho do software é inviável encontrar um valor médio de produtividade.

É importante mencionar também que, a partir do momento que se fundamenta a estimativa com medidas de tamanho e produtividade, tem-se mais embasamento em negociações com o cliente relativas ao prazo para entrega de produtos [McConnell, 2006].

Devido ao exposto fez-se necessária a procura de um modelo ou técnica de estimativa que cobrisse a maioria das atividades desenvolvidas para implementação do SDK, além de possibilitar a quantificação do software a ser desenvolvido a partir de parâmetros das APIs que possam ser razoavelmente estimados. Além de não perder a agilidade do tempo de resposta requerido em momentos de pré-venda (proposta) de um software.

Pensando nesses fatores e avaliando as técnicas de estimativa mais utilizadas no mercado, constatou-se que seria mais adequado o desenvolvimento de uma técnica específica para contagem do tamanho da API a ser desenvolvida para o SDK.

## 3. Descrição da metodologia

Nesta seção descreveremos a metodologia utilizada para desenvolvimento da técnica de estimativa que quantifica o tamanho de uma API, denominado API Points. A Figura 1 ilustra as etapas da metodologia utilizada.

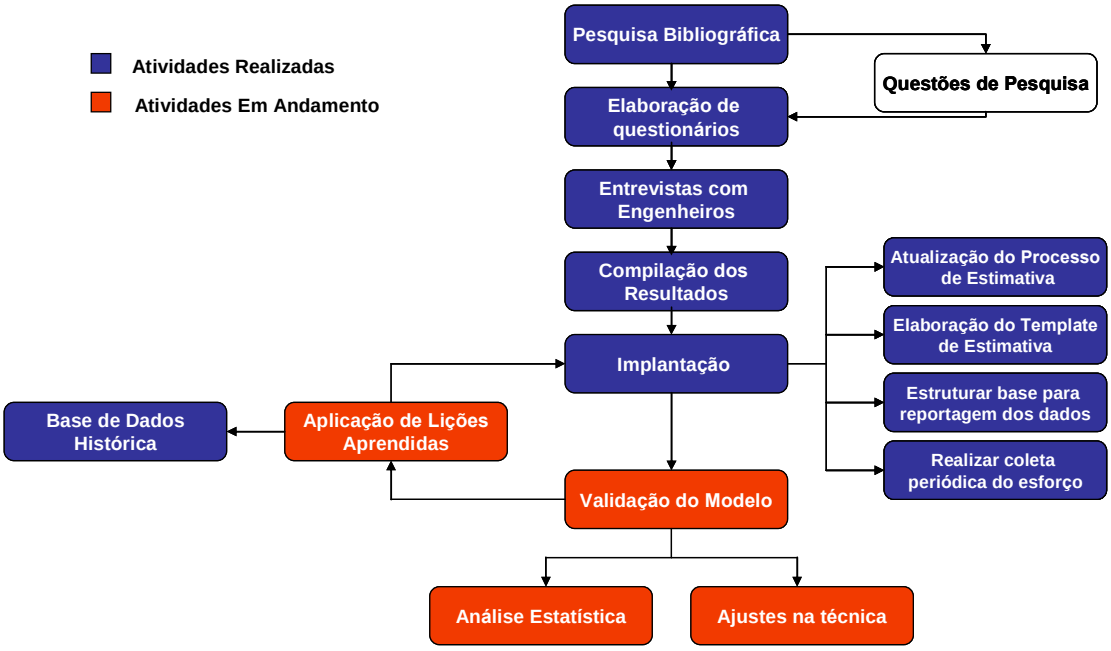


**Figura 1. Etapas da metodologia**

### 3.1. Pesquisa bibliográfica

Para identificar trabalhos relacionados, aprofundar conhecimento em estimativa de esforço para projetos de desenvolvimento de software e pesquisar por técnicas de estimativa existentes no mercado que pudessem suprir a necessidade do projeto.

### 3.2. Elaboração de questionários

Com o intuito de entrevistar os desenvolvedores do SDK a fim de identificar lacunas no método e no processo de estimativa, e aplicação de lições aprendidas para levantamento de issues e consequentemente identificação da *root cause* dos mesmos.

### 3.3. Compilação dos resultados

Através da Compilação dos Resultados dos questionários aplicados foi possível identificar que não existia uma padronização entre os desenvolvedores na seleção dos critérios estabelecidos para cálculo do esforço no desenvolvimento de uma API. Além disso, desenvolvedores iniciantes no projeto desconhecem uma série de fatores da arquitetura do sistema que, quando impactados, geram um esforço considerável de implementação.

### 3.4. Implantação

#### 3.4.1. Atualização do processo de estimativa

Como resultado desta atividade, o processo de estimativa do projeto foi atualizado incluindo atividades como “Escolher método de estimativa” e “Revisão / Inspeção das

Estimativas", "Atualização do processo e método" e "Aplicação de Lições Aprendidas focando em estimativa de tamanho e esforço". A Figura 2 ilustra o processo de estimativa utilizado pelo projeto atualmente.

O processo de estimativas pode ser melhorado a cada iteração com ajuda dos usuários do processo, com o resultado da coleta de lições aprendidas e construção da base de dados histórica.

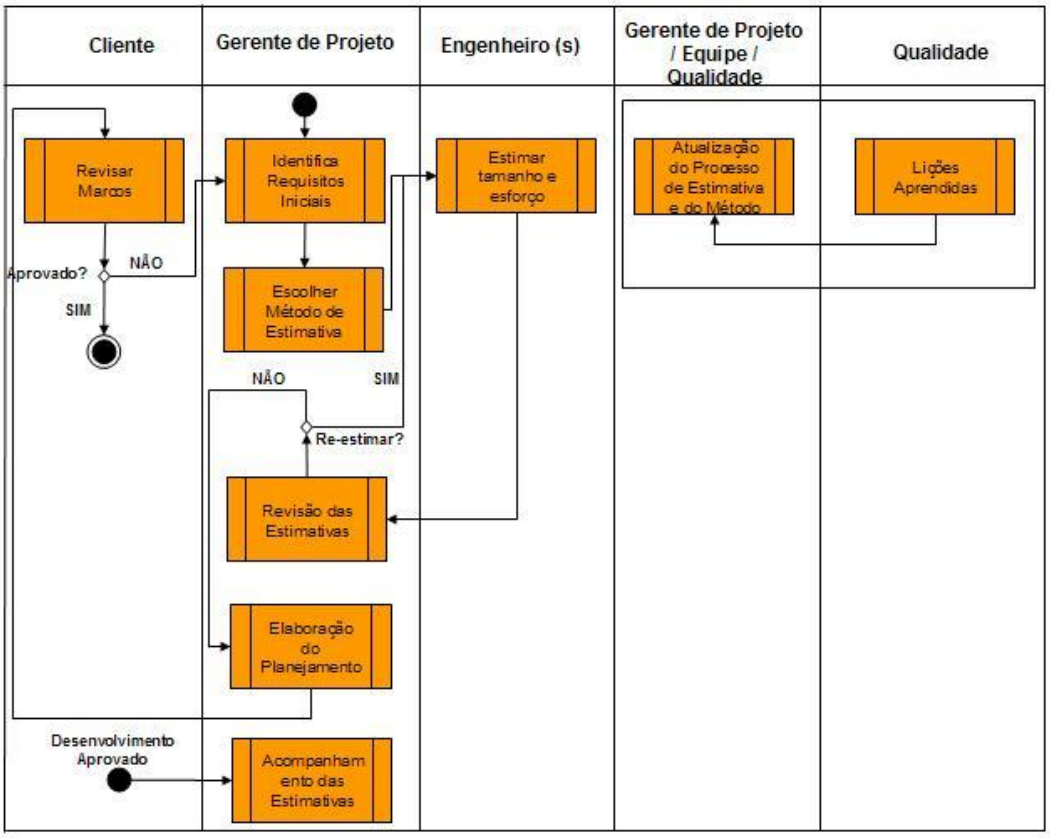


**Figura 3. Processo de Estimativa**

### 3.4.2. Elaboração do template

Elaboração do template para estimativa de tamanho de API contendo todos os fatores necessários para contagem do tamanho de uma API (ver Tabela 2).

### 3.4.3. Estruturar base para reportagem dos dados

Estruturar base para reportagem dos dados a fim de conseguir coletar o esforço realizado de forma padronizada em todo o projeto.

### 3.4.4. Realizar coleta periódica do esforço

O esforço que foi realizado ou despendido para cada trabalho será coletado e comparado com o esforço planejado.

### 3.5. Validação do modelo

#### 3.5.1. Análise estatística dos dados coletados

Esta atividade tem como objetivo analisar estatisticamente (regressão linear [Triola, 2008]) o resultado da coleta de esforço, verificando o grau de incidência de cada fator.

#### 3.5.2. Ajustes do método

Caso seja possível, serão realizadas adaptações no método de estimativa existente a fim de contemplar os novos fatores encontrados na etapa anterior ou ajuste do peso de um fator já existente.

### 3.6. Aplicação de Lições Aprendidas

A aplicação de Lições Aprendidas é extremamente importante para reflexão do que foi realizado e entendimento de causa raiz do issues mais críticos [Jorgensen, 2008]. O questionário de lições aprendidas foi adaptado para focar nas atividades de estimativa (processo e técnica) e planejamento do projeto.

### 3.7. Atualização da base de dados histórica

Será efetuada a atualização da base de dados histórica com esforço realizado nas implementações, restrições e riscos encontrados.

## 4. API Points

A técnica API Points desenvolvida pelo C.E.S.A.R [CESAR, 2008] em 2007, tem como foco estimar o tamanho e esforço necessário para desenvolvimento de uma API.

Esta técnica pode ser classificada como uma técnica de estimativa *Proxy Points* [Laird & Brennan, 2006], pois determina o tamanho do projeto, baseada em características externas do sistema, isto é características que podem ser utilizadas como um proxy para tamanho. O processo de estimativa utilizando API Points é composto por duas etapas:

### 4.1. Identificação das restrições

Fatores que possam limitar o desenvolvimento do projeto, premissas (suposições para propósito de planejamento são consideradas verdadeiras), dependências externas e internas, e riscos (detalhamento dos possíveis riscos e ações para mitigar seus efeitos durante o desenvolvimento do projeto) associados à API a ser desenvolvida.

### 4.2. Escolha do percentual de complexidade

A escolha do percentual de complexidade de cada fator conforme descrito na Tabela 2, variam numa escala de 0 a 4, de acordo com o grau de influência do fator em questão para implementação da API.

Em alguns casos o valor 0 pode indicar que o fator não está presente ou não é influente, 2, 3 influência média e o valor 4 indica influência significativa no processo.

Por fim, a estimativa do esforço total requerida para desenvolver o sistema, considerando as várias fases do ciclo de vida de desenvolvimento do software, é

derivada do tamanho do sistema, aplicando-se um fator de produtividade (homem-hora/API Points).

**Tabela 2. Fatores de complexidade**

| ID | Fator | Descrição | Peso |
|---|---|---|---|
| **F1** | Disponibilidade da documentação | A documentação disponível neste momento é suficiente para calcular este API? | 0.5 |
| **F2** | Número de métodos (javadoc) | Número de métodos do javadoc da API | 0.2 |
| **F3** | Integração com Servidor | O API precisa de integração com servidor? | 0.8 |
| **F4** | Emula *Features* de Hardware | A API precisa emular *feature* de hardware? | 1 |
| **F5** | Uso de componentes de terceiro | A API depende do uso de componentes de terceiros? | 0.5 |
| **F6** | Portabilidade | A API precisa ser implementada em mais de um sistema operacional? | 0.4 |
| **F7** | Emula serviços do S.O. do device | A API precisa emular algum serviço relativo ao sistema operacional do dispositivo? | 0.6 |
| **F8** | O processamento interno complexo | O processamento interno é complexo, ou seja, possui processamento lógico e matemático extensivo. | 0.8 |
| **F9** | Requisitos de segurança ou/e características especiais de segurança. | Esta API requer segurança como SSL ou código para criptografia? | 0.7 |
| **F10** | Interação com outras APIs | Esta API requer integração com outras APIs? | 0.8 |
| **F11** | Domínio da regra de negócio | Todas as pessoas que estão trabalhando no projeto estão familiarizadas e dominam detalhes técnicos do mesmo? | 0.7 |
| **F12** | Concorrência | Esta API precisa de processamento paralelo? | 0.7 |
| **F13** | Uso de mais uma linguagem de programação | Quais linguagens de programação serão utilizadas para implementar a API em questão? | 0.8 |

O método se baseia em 13 fatores identificados no projeto, avaliando a funcionalidade do SDK que esta sendo estimada, também definindo os seus níveis de influência.

Na Figura 3, é possível verificar o tamanho de mais de 10 APIs utilizando a técnica API Points.

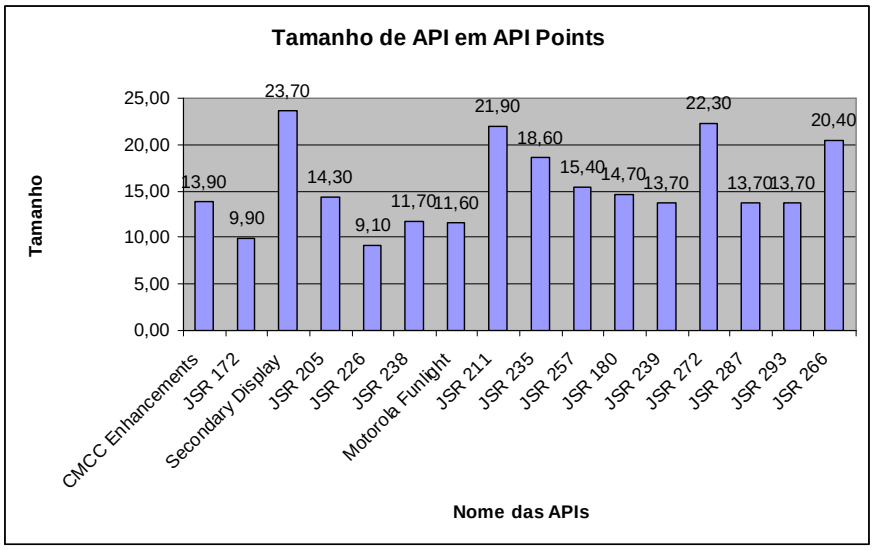


**Figura 3. Tamanho em API Points.**

## 5. Conclusão e trabalhos futuros

Uma das primeiras atividades que deve ser desempenhada antes de iniciar as etapas de processo de estimativa é entender o que se deseja estimar e então orientar a abordagem da estimativa. [Boehm & Fairley, 2000]

Já foi comprovado por dezenas de pesquisas que a melhoria no processo de estimativa se dá através da utilização de base de dados histórica de projetos organizacionais, de um processo sistematizado de estimativas [Vazquez, 2005], além da utilização de métodos de estimativa em conjunto com *checklist* que ajudam a reduzir a escala da incerteza de uma estimativa. As estimativas também precisam ser justificadas e comprovadas, pois apenas bons *feelings* não são justificativas aceitáveis. [Laird, 2006]

Com o início da implantação da técnica de API Points juntamente com atualização do processo de estimativa, espera-se conseguir desvios menores na estimativa de esforço e consequentemente planejamentos mais assertivo.

Como trabalho futuro deseja-se realizar análise estatística do grau de incidência de cada fator a fim de melhor calibrar a técnica através de regressão linear estatística [Triola, 2008] e enriquecer a base de dados histórica de projetos da organização.

## Referências Bibliográficas


Boehm, B.W. And Fairley,R.E. Software Estimation Perspectives. IEEE Software, Vol 17, Nº 6, November 2000, pp. 22-26.

C.E.S.A.R, Centro de Estudos e Sistemas Avançados de Recife, http://www.cesar.org.br/, visitado em 15/08/2008.

Chen, Z., Menzeis, T., Port, D., Boehm, B. Finding the Right Data for Software Cost Modeling. IEEE Software, 2005.

Java, Linguagem de Programação Java, http://www.java.com, visitado em 15/08/2008.

Jorgensen, M. And Gruschke, T. M. The impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments, 2008.

Laird, Linda M., The limitations of Estimation. Published by the IEEE Computer Society, 2006.

McConnell, Steven. Software Estimation – Demystifying the Black Art. Microsoft, 2006.

Mehwish, Nasir. A Survey of Software Estimation Techniques and Project Planning Pratices. IEEE, 2006.

Mulcahy, Rita. Preparatório para o Exame de PMP. 3ª Edição. RMC Publications, Inc, 2007.

PMBOK. Project management body of knowledge. Belo Horizonte: PMIMG, 2000.

Triola, Mario F. Introdução à Estatística. 10ª Edição – LTC, 2008.

Vazquez, C. E.; Simões, G. S; Albert, R. M. Análise de Pontos de Função – Medição, Estimativas e Gerenciamento de Projetos de Software. 3.ed. São Paulo: Editora Érica, 2005.