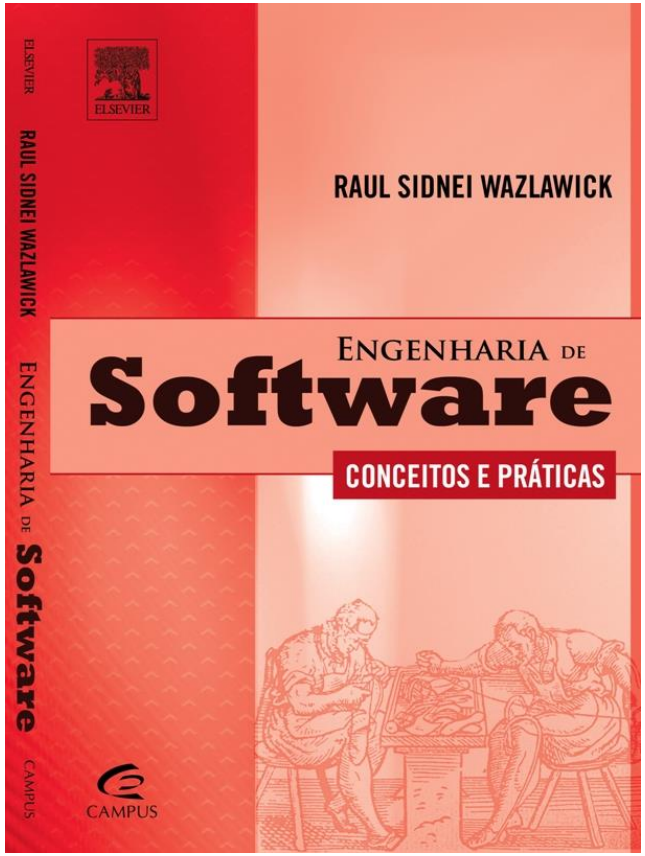


Estimação de Esforço

Engenharia de Software: Conceitos e Práticas

Prof. Raul Sidnei Wazlawick
UFSC-CTC-INE

Elsevier, 2013

Estimação de Esforço

- Uma das questões fundamentais em um projeto de software é saber, antes de executá-lo, quanto esforço, em horas de trabalho, será necessário para levá-lo a termo.

- Essa área, chamada de *estimativa de esforço* conta com algumas técnicas que têm apresentado resultados interessantes ao longo dos últimos anos.

- A maioria das técnicas de estimação de esforço utilizam pelo menos um parâmetro como base, por isso são chamadas de *técnicas paramétricas*.

SLOC e KSLOC

- A técnica conhecida como *LOC* (*Lines of Code*) ou *SLOC* (*Source Lines of Code*) foi possivelmente a primeira a surgir e consiste em estimar o número de linhas que um programa deverá ter, normalmente a partir da opinião de especialistas e histórico de projetos passados.

- Rapidamente a técnica evoluiu para a forma conhecida como *KSLOC* (*Kilo Source Lines of Code*), tendo em vista que o tamanho da maioria dos programas passou a ser medido em milhares de linhas.

Estimação de KSLOC

- Reunir a equipe para discutir o sistema a ser desenvolvido.
- Cada participante dará a sua opinião sobre a quantidade de KSLOC que serão necessárias para desenvolver o sistema.
- Usualmente a reunião não chegará a um valor único.
- Deverão então ser considerados pelo menos 3 valores:
  - O KSLOC ***otimista***, ou seja, o número mínimo de linhas que se espera desenvolver se todas as condições forem favoráveis.
  - O KSLOC ***pessimista***, ou seja, o número máximo de linhas que se espera desenvolver ante condições desfavoráveis.
  - O KSLOC **esperado**, ou seja, o número de linhas que efetivamente se espera desenvolver em uma situação de normalidade.

KSLOC usado nas estimativas

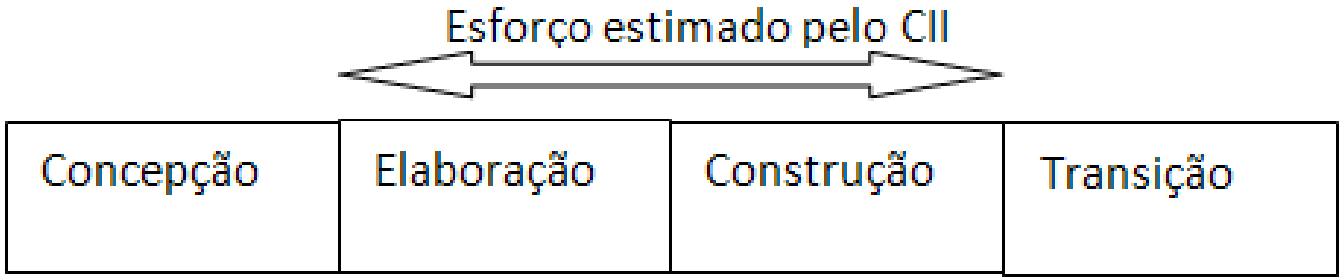
$$KSLOC = (4*KSLOC_{esperado} + KSLOC_{otimista} + KSLOC_{pessimista}) / 6$$

COCOMO

- *Constructive Cost Model* (também conhecido como COCOMO 81).
- Este modelo já é obsoleto, e foi substituído por COCOMO II em aplicações reais.
- O modelo COCOMO foi criado por Boehm (1981) a partir de um estudo empírico sobre sessenta e três projetos na empresa TRW Aerospace.
  - Os programas examinados tinham de 2 a 100 KSLOC e eram escritos em linguagens tão diversas quanto Assembly e PL/I.

COCOMO II

COCOMO II, ou *CII* é uma evolução do antigo modelo COCOMO 81 e ao contrário de seu antecessor, funciona bem com ciclos de vida iterativos e é fortemente adaptado para uso com o Processo Unificado, embora também seja definido para os modelos Cascata e Espiral.



Análise de Pontos de Função

- A técnica de *Análise Pontos de Função* (*APF*), ou *Function Point Analysis* (Albrecht & Gaffney Jr., 1983) é também uma técnica paramétrica para estimação de esforço para desenvolvimento de software.

- Porém, ao contrário de COCOMO, ela não se baseia em linhas de código, mas em requisitos.

- A análise de pontos de função é aplicável, portanto, a partir do momento em que os requisitos funcionais do software tenham sido definidos.

- Esses requisitos serão convertidos em valores numéricos, que depois de ajustados à capacidade da empresa desenvolvedora, representarão o esforço necessário para desenvolver o sistema.

- Assim, a medida obtida pela técnica é, a princípio, independente de linguagem de programação e de tecnologia empregada.

Três possíveis objetivos de contagem

- *Contagem para desenvolvimento de projeto.*
  - Esta técnica é usada para estimar o esforço para o desenvolvimento de um novo projeto.

- *Contagem para melhoria de projeto.*
  - Esta técnica é usada para evolução de software, onde se conta as funcionalidades adicionadas, alteradas e removidas.
  - A técnica é aplicável apenas para a manutenção adaptativa, já que a manutenção corretiva e perfectiva são muito imprevisíveis.

- *Contagem de aplicação.*
  - Esta técnica é usada para contar pontos de função de aplicações existentes.
  - Essa contagem pode ter vários objetivos, entre os principais, estimar o tamanho funcional da aplicação, de forma a relativizar outras métricas.
  - Por exemplo, pode ser mais realista conhecer o número de defeitos por ponto de função do que simplesmente o número de defeitos do software.

Padrões

- *IFPUG* (*International Function Point Users Group*)
  - No Brasil: *BFPUG* (*Brazilian Function Point Users Group*).
  - A técnica é reconhecida como métrica de software pela ISO na norma ISO/IEC 20926 – *Software Engineering – Funcion Point Counting Practices Manual*.

- NESMA, da associação holandesa de métricas.

- Mark II (Symons, 1988), ou MK II, mantido pela associação inglesa de métricas.
  - Ao contrário do manual de contagem do IFPUG, que deve ser adquirido, os manuais NESMA e MK II podem ser obtidos gratuitamente em seus *sites*, bastando fazer registro gratuito na respectiva associação.

Passos

- Determinar o tipo de contagem (desenvolvimento, melhoria ou aplicação existente).
- Determinar os limites da aplicação (escopo do sistema).
- Identificar e atribuir valor em pontos de função não ajustados para as transações sobre dados (entradas, consultas e saídas externas).
- Identificar e atribuir valor em pontos de função não ajustados (UFP) para os dados estáticos (arquivos internos e externos).
- Determinar o fator de ajuste técnico (VAF).
- Calcular o número de pontos de função ajustados (AFP).

Interpretação e Classificação dos Requisitos como Funções

- Apenas funcionalidades **visíveis para o usuário** devem ser contadas.

- Apenas transferências de informação para dentro e para fora do escopo do sistema (e arquivos de dados mantidos no sistema e acessíveis pelo usuário) são considerados funções.

A técnica APF avalia as duas naturezas dos dados:

- *Dados estáticos*,
  - ou seja, a representação estrutural dos dados, na forma de arquivos internos ou externos.

- *Dados dinâmicos*,
  - ou seja, a representação das transações sobre os dados, na forma em entradas, saídas e consultas externas.

Tipos de funções

- *Entradas externas*.
  - São entradas de dados ou controle, que tem como consequência a alteração do estado interno das informações do sistema.

- *Saídas externas*.
  - São saídas de dados que podem ser precedidas ou não da entrada de parâmetros.
  - Pelo menos um dos dados de saída deve ser derivado, ou seja, calculado.

- *Consultas externas*.
  - São saídas de dados que podem ser precedidas ou não da entrada de parâmetros.
  - Os dados devem sair da mesma forma como estavam armazenados, sem transformações ou cálculos.

- *Arquivo interno*.
  - É um elemento do modelo conceitual percebido pelo usuário e mantido internamente pelo sistema.

- *Arquivo externo*.
  - É um elemento do modelo conceitual percebido pelo usuário e mantido externamente por outras aplicações.

Parâmetros para estimar complexidade de funções

- *Registro* (*RET - Record Element Type*),
  - que corresponde a um subconjunto de dados reconhecível pelo usuário dentro de um arquivo interno ou externo (uma classe qualquer).

- *Arquivo* (*FTR - File Types Referenced*),
  - que corresponde a um arquivo interno ou externo, usado em uma transação (uma classe **que não seja componente de outra)**.

- *Argumento* (*DET - Data Element Type*),
  - que corresponde a uma unidade de informação (um campo), a princípio indivisível e reconhecível pelo usuário, normalmente seria um campo de uma tabela, um atributo de uma classe ou um parâmetro de uma função.

Tabela 7-38: Complexidade funcional de entradas externas.

	Argumentos DET		
Classes FTR	**1 a 4**	**5 a 15**	**16 ou mais**
0 a 1	Baixa	Baixa	Média
2	Baixa	Média	Alta
3 ou mais	Média	Alta	Alta

Tabela 7-39: Complexidade funcional de saídas e consultas.

	Argumentos DET		
Classes FTR	**1 a 5**	**6 a 19**	**20 ou mais**
0 a 1	Baixa	Baixa	Média
2 a 3	Baixa	Média	Alta
4 ou mais	Média	Alta	Alta

Tabela 7-40: Complexidade funcional de arquivos internos e externos.

	Argumentos DET		
Classes RET	**1 a 19**	**20 a 50**	**51 ou mais**
1	Baixa	Baixa	Média
2 a 5	Baixa	Média	Alta
6 ou mais	Média	Alta	Alta

Pontos de função não ajustados (UPF)

Tabela 7-41: Pontos de função não ajustados por tipo e complexidade de função.

	Complexidade funcional		
Tipo de função	**Baixa**	**Média**	**Alta**
Entrada	3	4	6
Saída	4	5	7
Consulta	3	4	6
Arquivo interno	7	10	15
Arquivo externo	5	7	10

Exemplo

- Requisitos:
  - O sistema deve permitir o gerenciamento (CRUDL) de informações sobre livros e usuários. Dos livros inclui-se: título, ISBN, autor, número de páginas, editora e ano de publicação. Dos usuários inclui-se: nome, documento, endereço, telefone e email.
  - O sistema deve permitir o registro de empréstimos onde são informados o documento do usuário e o ISBN de cada um dos livros.
  - Quando um empréstimo for executado, o sistema deve armazenar as informações em uma tabela relacional usando chaves estrangeiras para identificar o usuário e os livros.
  - Após o registro de um empréstimo deve ser impresso um recibo com o nome do usuário, e título e data de devolução prevista para cada livro que deve ser calculada como a data atual somada ao prazo do livro.

Tabela 7-42: Exemplo de identificação de funções a partir de requisitos.

Função	Tipo	FTR	RET	DET	#FTR	#RET	#DET	Complex.	UFP
Cadastro de Livros	Arquivo interno		Livro, Editora, Autor	título, isbn, autor, número de páginas, editora, ano de publicação		3	6	Baixa	7
• Inserir Livro	Entrada externa	Livro, Editora, Autor		título, isbn, autor, número de páginas, editora, ano de publicação	3		6	Alta	6
• Alterar Livro	Entrada externa	Livro, Editora, Autor		título, isbn, autor, número de páginas, editora, ano de publicação	3		6	Alta	6
• Excluir Livro	Entrada externa	Livro		título, isbn, ano de publicação	1		3	Baixa	3
• Consultar Livro	Consulta externa	Livro, Editora, Autor		título, isbn, autor, número de páginas, editora, ano de publicação	3		6	Média	4
• Listar Livros	Consulta externa	Livro, Editora, Autor		título, isbn, autor, número de páginas, editora, ano de publicação	3		6	Média	4

Cadastro de Usuários	Arquivo interno		Pessoa	nome, documento, endereço, telefone, email		1	5	Baixa	7
• Inserir Usuário	Entrada externa	Pessoa		nome, documento, endereço, telefone, email	1		5	Baixa	3
• Alterar Usuário	Entrada externa	Pessoa		nome, documento, endereço, telefone, email	1		5	Baixa	3
• Excluir Usuário	Entrada externa	Pessoa		nome, documento	1		2	Baixa	3
• Consultar Usuário	Consulta externa	Pessoa		nome, documento, endereço, telefone, email	1		5	Baixa	3
• Listar Usuários	Consulta externa	Pessoa		nome, documento	1		2	Baixa	3
Registrar empréstimo	Entrada externa	Pessoa, Livro, Emprestimo		nome da pessoa, isbn dos livros	3		2	Média	4
Imprimir recibo	Consulta externa	Pessoa, Livro, Emprestimo		nome da pessoa, título do livro, data de hoje, prazo do livro, data de devolução	3		5	Baixa	3
TOTAL									**59**

AFP – Pontos de Função Ajustados

- O método de pontos de função não tem fatores de escala como COCOMO.
- Então ele presume que o esforço será linear em relação à quantidade de funcionalidades implementadas.
- Porém, o método possui um conjunto de fatores de ajuste técnico já que diferentes projetos e diferentes equipes poderão produzir funcionalidades em ritmos diferentes.

A técnica de pontos de função sugere 14 fatores de ajuste técnico, conhecidos como GSC(*General Systems Characteristics*)

- Comunicação de dados.
- Processamento de dados distribuído.
- Performance.
- Uso do sistema.
- Taxa de transações.
- Entrada de dados online.
- Eficiência do usuário final.
- Atualização online.
- Processamento complexo.
- Reusabilidade.
- Facilidade de instalação.
- Facilidade de operação.
- Múltiplos locais.
- Facilidade para mudança.

A avaliação dos GSC é feita para o projeto como um todo (não para cada função). Então, como são 14 GSC e cada um receberá uma nota de 0 a 5, a somatória total das notas ficará entre 0 e 70. Esse valor ajustado é conhecido como *VAF* (*Value Adjustment Factor*):

$$VAF = 0{,}65 + \left(0{,}01 * \sum_{i=1}^{14} GSC_i\right)$$

Assim, o somatório dos *GSC* multiplicado por 0,01 e somado a 0,65 vai fazer com que *VAF* varie de 0,65 a 1,35.

Esse valor é multiplicado pelo número de UFP para obter o AFP, ou número de pontos de função ajustados:

$$AFP = UFP * VAF$$

Assim, em um projeto onde todos os fatores técnicos sejam mínimos (nota 0), o *AFP* será igual a 65% do valor nominal de *UFP*. Já em um sistema onde todos os fatores técnicos sejam máximos (nota 5) o *AFP* será igual a 135% do valor nominal de *UFP*. Um sistema nominal seria aquele onde todos os fatores técnicos tenham nota 3, o que levaria o *AFP* a ser igual ao *UFP*.

Detalhamento dos fatores técnicos

- Comunicação de dados.
- Processamento de dados distribuído.
- Performance.
- Uso do sistema.
- Taxa de transações.
- Entrada de dados *online*.
- Eficiência do usuário final.
- Atualização *online*.
- Processamento complexo.
- Reusabilidade.
- Facilidade de instalação.
- Facilidade de operação.
- Múltiplos locais.
- Facilidade para mudança.

Entrada de dados online

- Avalia a percentagem de informação que o sistema deve obter *online*, ou seja, dos usuários em tempo real:
  - 0: todas as transações são processadas em modo *batch*.
  - 1: 1% a 7% das transações são entradas de dados interativas.
  - 2: 8% a 15% das transações são entradas de dados interativas.
  - 3: 16% a 23% das transações são entradas de dados interativas.
  - 4: 24% a 30% das transações são entradas de dados interativas.
  - 5: mais de 30% das transações são entradas de dados interativas.

Pontos de Caso de Uso

- A técnica de *Pontos de Caso de Uso* surgiu em 1993 a partir da Tese de Gustav Karner (1993).

- O método é baseado em Análise de Pontos de Função, especificamente MK II, que é um modelo relativamente mais simples que o do IFPUG.

- O método se baseia na análise da quantidade e complexidade dos atores e casos de uso, o que gera os UUCP, ou pontos de caso de uso não ajustados. Depois, a aplicação e fatores técnicos e ambientais leva aos UCP, ou pontos de caso de uso (ajustados).

UAW - Complexidade de Atores

- O valor de *UAW* (*Unajusted Actor Weight*) é a soma dos pontos atribuídos a todos os atores relacionados no sistema:
  - *Atores humanos que interagem com o sistema através de interface gráfica* são considerados complexos e recebem 3 pontos de caso de uso.
  - *Sistemas que interagem por um protocolo como TCP/IP* e *atores humanos que interagem com o sistema apenas por linha de comando* são considerados de média complexidade, e recebem 2 pontos de caso de uso.
  - *Sistemas que são acessados por interfaces de programação* (*API*) são considerados de baixa complexidade, e recebem 1 ponto de caso de uso.

UUCW – Complexidade dos Casos de Uso

- O valor de *UUCW* (*Unajusted Use Case Weight*) é dado pela soma dos valores atribuídos a cada um dos casos de uso da aplicação.

- Na proposta original de Karner, a complexidade de um caso de uso era definida em função do número estimado de transações (movimentos de informação para dentro ou para fora do sistema), incluindo as sequências alternativas do caso de uso:
  - Casos de uso *simples* devem possuir no máximo 3 transações, e recebem 5 pontos de caso de uso.
  - Casos de uso *médios* devem possuir de 4 a 7 transações, e recebem 10 pontos de caso de uso.
  - Casos de uso *complexos* devem possuir mais de 7 transações, e recebem 15 pontos de caso de uso.

- Uma forma alternativa de estimar a complexidade de um caso de uso é em função da quantidade de classes necessária para implementar as funções do caso de uso:
  - Casos de uso simples devem ser implementados com 5 classes ou menos.
  - Casos de uso médios devem ser implementados com 6 a 10 classes.
  - Casos de uso complexos devem ser implementados com mais de 10 classes.

- Outra forma ainda de estimar a complexidade de um caso de uso é pela análise de seu risco. Assim:
  - Casos de uso como relatórios, têm apenas uma ou duas transações e baixo risco, pois não alteram dados, e podem ser considerados casos de uso simples.
  - Casos de uso padronizados, como CRUD, têm um número conhecido e limitado de transações, têm médio risco (pois embora a lógica de funcionamento seja conhecida, regras de negócio obscuras podem existir), e podem ser considerados como casos de uso médios.
  - Casos de uso não padronizados têm um número desconhecido de transações e alto risco, pois além das regras de negócio serem desconhecidas, ainda deve-se descobrir qual é o fluxo principal e quais as sequências alternativas. Assim, esse tipo de caso de uso deverá ser considerado como complexo.

UUCP – Pontos de Caso de Uso não Ajustados

- O valor de pontos de caso de uso não ajustados, ou UUCP, é calculado simplesmente como:

$$UUCP = UAW + UUCW$$

TCF - Fatores Técnicos

- Pontos de caso de uso fazem o ajuste dos pontos em função de dois critérios: fatores técnicos (que pertencem ao projeto) e fatores ambientais (que pertencem à equipe).

- Cada fator recebe uma nota de 0 a 5, onde 0 indica nenhuma influência do projeto, 3 é a influência nominal e 5 máxima influência no projeto.

Tabela 7-43: Fatores técnicos de ajuste de pontos de caso de uso.

Sigla	Fator	Peso
T1	Sistema Distribuído	2
T2	Performance	2
T3	Eficiência de usuário final	1
T4	Complexidade de processamento	1
T5	Projeto visando código reusável	1
T6	Facilidade de instalação	0,5
T7	Facilidade de uso	0,5
T8	Portabilidade	2
T9	Facilidade de mudança	1
T10	Concorrência	1
T11	Segurança	1
T12	Acesso fornecido a terceiros	1
T13	Necessidades de treinamento	1

$$TCF = 0{,}6 + (0{,}01 * TFactor)$$

EF – Fatores Ambientais

- Um aspecto que distingue a técnica de pontos de caso de uso de pontos de função e CII é que ela tem um fator de ajuste específico para as características da equipe de desenvolvimento.

- Assim, pode-se tomar o mesmo projeto, com os mesmos fatores técnicos, e ele poderá ter pontos de caso de uso ajustados diferentes para equipes diferentes.

Tabela 7-44: Fatores ambientais de ajuste de pontos de caso de uso.

Sigla	Fator	Peso
E1	Familiaridade com o processo de desenvolvimento	1,5
E2	Experiência com a aplicação	0,5
E3	Experiência com orientação a objetos	1
E4	Capacidade do analista líder	0,5
E5	Motivação	1
E6	Estabilidade de requisitos obtida historicamente	2
E7	Equipe em tempo parcial	-1
E8	Dificuldade com a linguagem de programação	-1

$$EF = 1{,}4 - (0{,}03 * EFactor)$$

UCP – Pontos de Caso de Uso Ajustados

$$UCP = UUCP * TCF * EF$$

Estimativas baseada em casos de uso – PCU

Exemplo

Pontos por Caso de Uso

Trata de estimar o tamanho de um sistema de acordo com:

- o modo como os usuários o utilizarão;
- a complexidade de ações requerida por cada tipo de usuário;
- uma análise em alto nível dos passos necessários para a realização de cada tarefa;

Pontos por Caso de Uso

O Método de Use Case Points foi criado para que seja possível estimar o tamanho de um sistema já na fase de levantamento de Casos de Uso;

Ele utiliza-se dos próprios documentos gerados nesta fase de análise como subsídio para o cálculo dimensional;

Estudo de Caso

Sistema que será usado como exemplo:

- Site de suporte de produtos para uma grande companhia de software;
- A estimativa foi feita a partir dos casos de uso de nível muito alto (*business modelling*), que foram criados em tempo de levantamento de requisitos;
- Os atores identificados foram 4 usuários humanos diferentes usando uma interface gráfica nesses casos de uso;

Estudo de Caso – continuação

Foram identificados 23 casos de uso sendo que destes 7 eram simples, 13 eram médio e 3 eram complexos.

Quanto aos fatores técnicos foram levantados: Sistema distribuído (1), Tempo de resposta, Eficiência, Processamento complexo e Facilidade de mudança (3), Facilidade de uso (5), e o restante não teve influência nenhuma (0).

Quanto aos fatores ambientais foram levantados: Familiaridade com metodologia, Experiência em Orientação a Objetos, Presença de analista experiente e Motivação (5), Requisitos estáveis (3) e o restante não teve influência nenhuma (0).

Pontos por Caso de Uso

Passo 1: Cálculo do UAW (*Unadjusted Actor Weight*)

Tipo de Ator	Peso	Descrição
Ator Simples	1	Outro sistema acessado através de uma API de programação
Ator Médio	2	Outro sistema acessado interagindo através da rede
Ator Complexo	3	Um usuário interagindo através de uma interface gráfica

Pontos por Caso de Uso

No caso do exemplo:

Tipo de Ator	Peso	Nº de atores	Resultado
Ator Simples	1	0	0
Ator Médio	2	0	0
Ator Complexo	3	4	12
		Total UAW	**12**

Valores já calculados: **UAW = 12**

Pontos por Caso de Uso

Passo 2: Cálculo do UUCW (*Unadjusted Use Case Weight*)

UUCW – Complexidade dos Casos de Uso

O valor de *UUCW* (*Unajusted Use Case Weight*) é dado pela soma dos valores atribuídos a cada um dos casos de uso da aplicação.

Na proposta original de Karner, a complexidade de um caso de uso era definida em função do número estimado de transações (movimentos de informação para dentro ou para fora do sistema), incluindo as sequências alternativas do caso de uso:

- Casos de uso *simples* devem possuir no máximo 3 transações, e recebem 5 pontos de caso de uso.
- Casos de uso *médios* devem possuir de 4 a 7 transações, e recebem 10 pontos de caso de uso.
- Casos de uso *complexos* devem possuir mais de 7 transações, e recebem 15 pontos de caso de uso.

Uma forma alternativa de estimar a complexidade de um caso de uso é em função da quantidade de classes necessária para implementar as funções do caso de uso:

- Casos de uso simples devem ser implementados com 5 classes ou menos.
- Casos de uso médios devem ser implementados com 6 a 10 classes.
- Casos de uso complexos devem ser implementados com mais de 10 classes.

Outra forma ainda de estimar a complexidade de um caso de uso é pela análise de seu risco. Assim:

- Casos de uso como relatórios, têm apenas uma ou duas transações e baixo risco, pois não alteram dados, e podem ser considerados casos de uso simples.
- Casos de uso padronizados, como CRUD, têm um número conhecido e limitado de transações, têm médio risco (pois embora a lógica de funcionamento seja conhecida, regras de negócio obscuras podem existir), e podem ser considerados como casos de uso médios.
- Casos de uso não padronizados têm um número desconhecido de transações e alto risco, pois além das regras de negócio serem desconhecidas, ainda deve-se descobrir qual é o fluxo principal e quais as sequências alternativas. Assim, esse tipo de caso de uso deverá ser considerado como complexo.

Pontos por Caso de Uso

No caso do exemplo:

Tipo	Peso	Nº de Casos de Uso	Resultado
Simples	5	7	35
Médio	10	13	130
Complexo	15	3	45
		Total UUCW	**210**

Valores já calculados: **UAW = 12, UUCW = 210**

Pontos por Caso de Uso

Passo 3: Cálculo do UUCP (*Unadjusted Use Case Points*)

UUCP = UAW + UUCW

No caso do exemplo:

UUCP = 12 + 210 = 222

Valores já calculados: **UAW = 12, UUCW = 210, UUCP = 222**

Pontos por Caso de Uso

Calculando fatores de ajuste:

- O método de ajuste é bastante similar ao adotado pela Análise por Pontos de Função e é constituído de duas partes:
  - **Cálculo de fatores técnicos:** cobrindo uma série de requisitos funcionais do sistema;
  - **Cálculo de fatores de ambiente:** requisitos não-funcionais associados ao processo de desenvolvimento;

Pontos por Caso de Uso

Passo 4: Cálculo do Tfactor

- Para cada requisito listado na tabela, deve ser atribuído um valor que determina a influência do requisito no sistema, variando entre 0 e 5;

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

Pontos por Caso de Uso

No caso do exemplo:

Fator	Requisito	Peso	Influência	Resultado
T1	Sistema distribuído	2	1	2
T2	Tempo de resposta	2	3	6
T3	Eficiência	1	3	3
T4	Processamento complexo	1	3	3
T5	Código reusável	1	0	0
T6	Facilidade de instalação	0.5	0	0
T7	Facilidade de uso	0.5	5	2.5
T8	Portabilidade	2	0	0
T9	Facilidade de mudança	1	3	3
T10	Concorrência	1	0	0
T11	Recursos de segurança	1	0	0
T12	Acessível por terceiros	1	0	0
T13	Requer treinamento especial	1	0	0
			Tfactor	**19,5**

Pontos por Caso de Uso

Passo 5: Cálculo do TCF (Technical Complexity Factor)

$$\mathbf{TCF = 0.6 + (0.01 \times Tfactor)}$$

No caso do exemplo:

$$\mathbf{TCF = 0.6 + (0.01 \times 19.5) = 0.795}$$

Valores já calculados: **UUCP = 222, Tfactor = 19.5, TCF = 0.795**

Pontos por Caso de Uso

Passo 6: Cálculo do Efactor

- Para cada requisito listado na tabela, deve ser atribuído um valor que determina a influência do requisito no sistema, variando entre 0 e 5;

Fator	Descrição	Peso
E1	Familiaridade com RUP ou outro processo formal	1.5
E2	Experiência com a aplicação em desenvolvimento	0.5
E3	Experiência em Orientação a Objetos	1
E4	Presença de analista experiente	0.5
E5	Motivação	1
E6	Requisitos estáveis	2
E7	Desenvolvedores em meio-expediente	-1
E8	Linguagem de programação difícil	-1

Pontos por Caso de Uso

No caso do exemplo:

Fator	Descrição	Peso	Influência	Resultado
E1	Familiaridade com RUP ou outro processo formal	1.5	5	7.5
E2	Experiência com a aplicação em desenvolvimento	0.5	0	0
E3	Experiência em Orientação a Objetos	1	5	5
E4	Presença de analista experiente	0.5	5	2.5
E5	Motivação	1	5	5
E6	Requisitos estáveis	2	3	6
E7	Desenvolvedores em meio-expediente	-1	0	0
E8	Linguagem de programação difícil	-1	0	0
			Efactor	**26**

Valores já calculados: **UUCP = 222, TCF = 0.795, Efactor = 26**

Pontos por Caso de Uso

Passo 7: Cálculo do ECF (Environmental Complexity Factor)

$$\mathbf{ECF = 1.4 + (-0.03 \times Efactor)}$$

No caso do exemplo:

$$\mathbf{ECF = 1.4 + (-0.03 \times 26) = 0.62}$$

Valores já calculados: **UUCP = 222, TCF = 0.795, Efactor = 26, ECF = 0.62**

Pontos por Caso de Uso

Passo 8: Cálculo dos UCP (Use Case Points)

$$\mathbf{UCP = UUCP \times TCF \times ECF}$$

No caso do exemplo:

ECF = 222 × 0.795 × 0.62 = 109.42 ou 109 Use Case Points

Valores já calculados: **UUCP = 222, TCF = 0.795, ECF = 0.62**

Pontos por Caso de Uso

Passo 9: Cálculo do tempo de trabalho estimado

- Para simplificar, utilizaremos a média de 20 horas por Ponto de Casos de Uso

No caso do exemplo:

Tempo estimado = 109 * 20 = **2180 horas de trabalho**