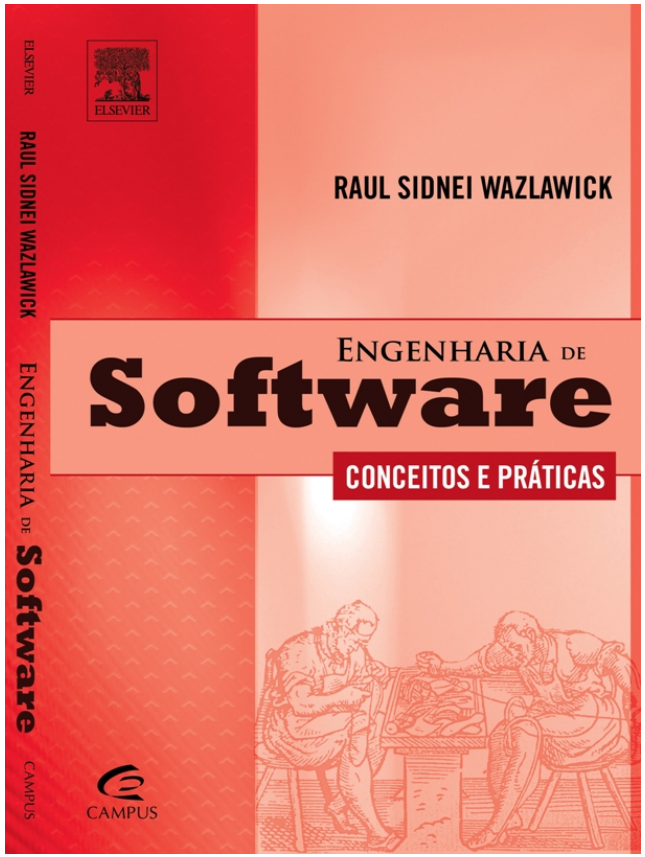


Estimação de Esforço

Engenharia de Sofware: Conceitos e Práticas

Prof. Raul Sidnei Wazlawick
UFSC-CTC-INE

Elsevier, 2013

Conteúdo

- SLOC e KSLOC
- COCOMO
- COCOMO II
- Pontos de Função
- Pontos de Caso de Uso
- Pontos de História

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

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

Backfire Tables (transformando pontos de função em KSLOC)

Linguagem	Default SLOC/UFP	Linguagem	Default SLOC/UFP
Access	38	Jovial	107
Ada 83	71	Lisp	64
Ada 95	49	Código de Máquina	640
AI Shell	49	Modula 2	80
APL	32	Pascal	91
Assembly – básico	320	Perl	27
Assembly – macro	213	PowerBuilder	16
Basic – ANSI	64	Prolog	64
Basic compilado	91	Query	13
C	128	Gerador de relatórios	80
C++	55	Linguagem de simulação	46
Cobol	91	Planilha	6
Forth	64	Scripts da Shell do Unix	107
HTML 3.0	15	Visual Basic	29
Java	53	Visual C++	34

Como contar KSLOC

- Em relação ao tipo de comando devem ser contados:
  - Comandos executáveis.
  - Declarações.
  - Diretivas de compilação.

- Por outro lado, não devem ser contados:
  - Comentários.
  - Linhas em branco.

- Em relação à forma como o código é produzido, devem ser contadas as linhas:
  - Programadas.
  - Copiadas ou reusadas.
  - Modificadas.

- Não devem ser contadas linhas:
  - Geradas por geradores automáticos de código.
  - Removidas.

- Em relação aos comandos presentes na maioria das linguagens de programação, devem ser contados:
  - Comandos *null*, *continue* e *no-op*.
  - Comandos que instanciam elementos genéricos.
  - Pares *begin-end* ou {...} usados em comandos estruturados.
  - Comandos *elseif*.
  - Palavras chave como *division*, *interface* e *implementation*.

- Não devem ser contados:
  - Comandos vazios como "; ", quando colocados sozinhos em uma linha.
  - Pares *begin-end* ou {...} usados em para delimitar o bloco principal ou procedimentos e funções.
  - Expressões passadas como argumentos para chamadas de procedimentos ou função (conta-se apenas a chamada).
  - Expressões lógicas em comandos IF, WHILE ou REPEAT (conta-se apenas o comando que contém as expressões).
  - Símbolos que servem como finalizadores de comandos executáveis, declarações ou subprogramas.
  - Símbolos THEN, ELSE e OTHERWISE, quando aparecerem sozinhos em uma linha (mas conta-se o comando que os sucede).

- São entendidos como *comandos executáveis*, e devem ser contados, todos os comandos que sejam atribuições, *GOTO*, chamada de procedimento, chamada de macro, retorno, *break*, *exit*, *stop*, *continue*, *null*, *noop*, etc. Também devem ser contadas separadamente as estruturas de controle como estruturas de repetição e seleção, e inclusive seus blocos *begin-end* ou {...}, se exisitirem, contam separadamente.

Exemplo

```
BEGIN                                          0
  IF (a <> b) OR (b < c) THEN                 +1
    BEGIN                                     +1
      X := b;                                 +1
    END                                        0
  ELSE                                         0
    X := fatorial(a)                          +1
  ;                                            0
END;                                           0
                                              ---
                                               4
```

- As declarações são comandos não executáveis que também devem ser contados.

- Por exemplo, declarações de nomes, números, constantes, objetos, tipos, subtipos, programas, subprogramas, tarefas, exceções, pacotes, genéricos e macros.

- Blocos *begin-end* ou {...} quando são parte obrigatória da declaração de subprogramas, não devem ser contados separadamente (apenas a declaração do subprograma conta).

Exemplo

```
procedure fib(num:integer):integer;                +1
var ultimoFib, cont, penultimoFib : integer;       +1
begin                                               0
   if num = 0 then                                 +1
     fib := 0                                      +1
   else                                             0
     if num = 1 then                               +1
       fib := 1                                    +1
     else                                           0
       Begin                                       +1
         ultimoFib := 1;                           +1
         cont := 1;                                +1
         repeat                                    +1
           penultimoFib := ultimoFib;              +1
           ultimoFib := fib;                       +1
           fib := penultimoFib + ultimoFib;        +1
           cont := cont + 1;                       +1
         until cont = num;                         +1
       end                                          0
     ;                                              0
   ;                                                0
end;                                                0
                                                   ----
                                                    15
```

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.

Implementações

- *Implementação básica*,
  - quando a única informação sobre o sistema efetivamente disponível é o número estimado de linhas de código.

- *Implementação intermediária*,
  - quando certos fatores relativos ao produto, suporte computacional, pessoal e processo são conhecidos e podem ser avaliados para o sistema a ser produzido.

- *Implementação avançada*,
  - quando for necessário subdividir o sistema em subsistemas e distribuir as estimativas de esforço por fase e atividade.

Tipos de projeto

- *Modo orgânico*,
  - que se aplica quando o sistema a ser desenvolvido não envolver dispositivos de hardware e a equipe estiver acostumada a desenvolver este tipo de aplicação, ou seja, sistemas de baixo risco tecnológico e baixo risco de pessoal.

- *Modo semidestacado*,
  - que se aplica a sistemas com maior grau de novidade para a equipe e que envolvem interações significativas com hardware, mas para os quais a equipe ainda tem algum conhecimento, ou seja, sistemas onde a combinação do risco tecnológico e de pessoal seja médio.

- *Modo embutido*,
  - que se aplica a sistemas com alto grau de interação com diferentes dispositivos de hardware, ou que sejam embarcados, e para os quais a equipe tenha considerável dificuldade de abordagem. São os sistemas com alto risco tecnológico e/ou de pessoal.

As três implementações do modelo COCOMO permitem determinar 3 informações básicas:

- O esforço estimado em desenvolvedor-mês: *E*.
- O tempo linear de desenvolvimento sugerido em meses corridos: *T*.
- O número médio de pessoas recomendado para a equipe: *P*.

Esforço, Tempo linear e Tamanho de equipe

$E = ab * KSLOC^{bb}$

$T = cb * E^{db}$

$P = E/T$

Tipo de projeto	*ab*	*bb*	*cb*	*db*
Orgânico	2,4	1,05	2,5	0,38
semidestacado	3,0	1,12	2,5	0,35
Embutido	3,6	1,2	2,5	0,32

COCOMO Intermediário

Tabela 7-4: Fatores influenciadores de custo do modelo COCOMO intermediário.

Fatores influenciadores de custo	Acrônimo	Muito baixa	Baixa	Média	Alta	Muito Alta	Extra Alta
Relativos ao produto							
Nível de confiabilidade requerida	RELY	0,75	0,88	1,00	1,15	1,40	
Dimensão da base de dados	DATA		0,94	1,00	1,08	1,16	
Complexidade do produto	CPLX	0,70	0,85	1,00	1,15	1,30	1,65
Suporte computacional							
Restrições ao tempo de execução	TIME			1,00	1,11	1,30	1,66
Restrições ao espaço de armazenamento	STOR			1,00	1,06	1,21	1,56
Volatilidade da máquina virtual	VIRT		0,87	1,00	1,15	1,30	
Tempo de resposta do computador	TURN		0,87	1,00	1,07	1,15	
Pessoal							
Capacidade dos analistas	ACAP	1,46	1,19	1,00	0,86	0,71	
Experiência no domínio da aplicação	AEXP	1,29	1,13	1,00	0,91	0,82	
Capacidade dos programadores	PCAP	1,42	1,17	1,00	0,86	0,70	
Experiência na utilização da máquina virtual	VEXP	1,21	1,10	1,00	0,90		
Experiência na linguagem de programação	LEXP	1,14	1,07	1,00	0,95		
Processo							
Adoção de boas práticas de programação	MODP	1,24	1,10	1,00	0,91	0,82	
Uso de ferramentas atualizadas	TOOL	1,24	1,10	1,00	0,91	0,83	
Histórico de projetos terminados no prazo	SCED	1,23	1,08	1,00	1,04	1,10	

Equações

- $EAF = RELY * DATA * CPLX * TIME * \ldots * SCED$
- $E = ai * KSLOC^{bi} * EAF$

Tabela 7-5: Valores de *ai* e *bi* em função do tipo de projeto.

Tipo de projeto	*ai*	*bi*
orgânico	2,8	1,05
semidestacado	3,0	1,12
embutido	3,2	1,2

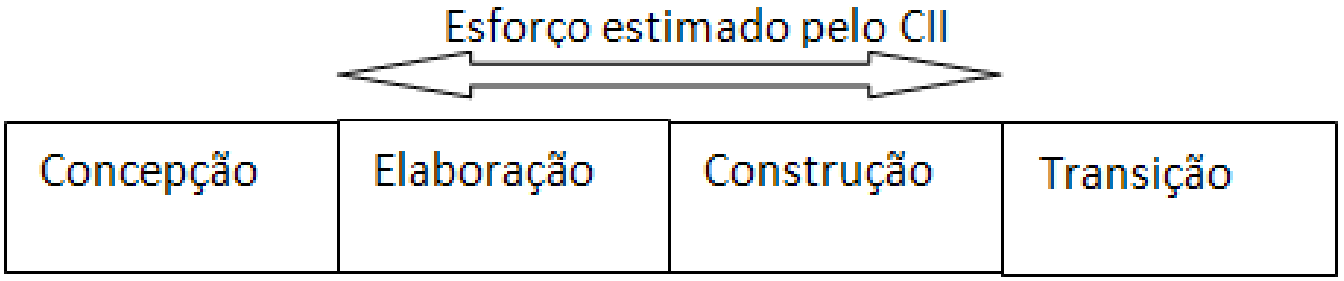
COCOMO Avançado

- A implementação avançada ou completa do modelo COCOMO introduz facetas como a decomposição do projeto em subprojetos, bem como estimativas individualizadas para as fases do projeto.

- Porém, como o modelo é complexo e desatualizado, sua apresentação será omitida, em função da apresentação de seu sucessor COCOMO II.

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.



Equação geral CII

$$E = A * KSLOC^S * \prod_{i=1}^{n} M_i$$

- *E* é o esforço total nominal que se deseja calcular para o projeto (fases de elaboração e construção).
- *A* é uma constante que deve ser calibrada a partir de dados históricos. CII sugere um valor inicial de 2,94.
- *KSLOC* é o número estimado de milhares de linhas de código que deverão ser desenvolvidas.
- *S* é o **coeficiente de esforço**, cujo cálculo é mostrado abaixo.
- M_i são os **multiplicadores de esforço**.

Coeficiente de Esforço

$$S = B + 0{,}001 * \sum_{j=1}^{5} F_j$$

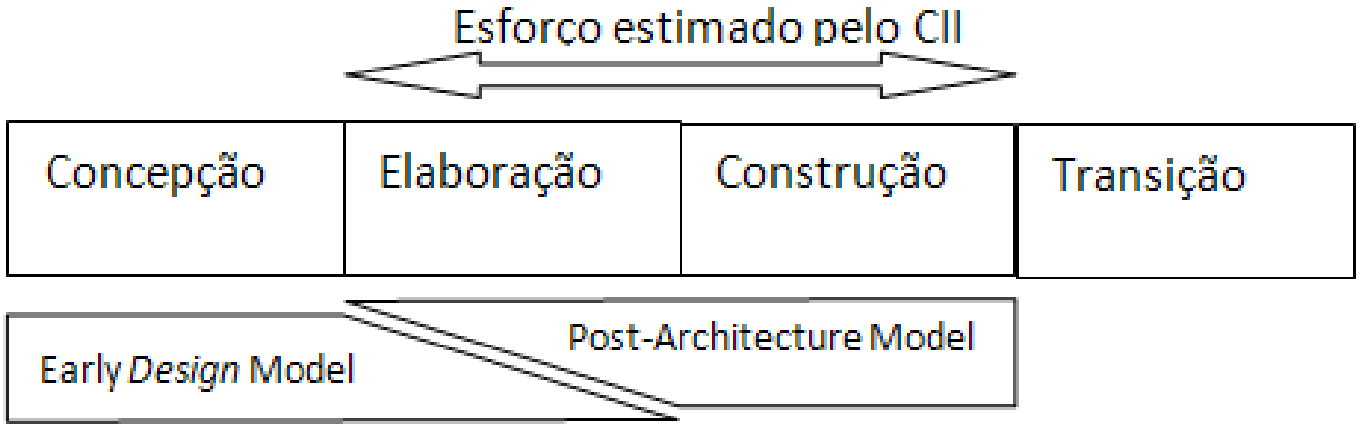
Onde:

a) S é o expoente de esforço que se deseja calcular.
b) B é uma constante que deve ser calibrada de acordo com valores históricos. CII sugere um valor inicial de 0,91.
c) F_j são cinco fatores de escala (*scale factors*) que devem ser atribuídos para cada projeto específico, tomando-se sempre muito cuidado pois sua influência no cálculo do esforço total do projeto é exponencial.

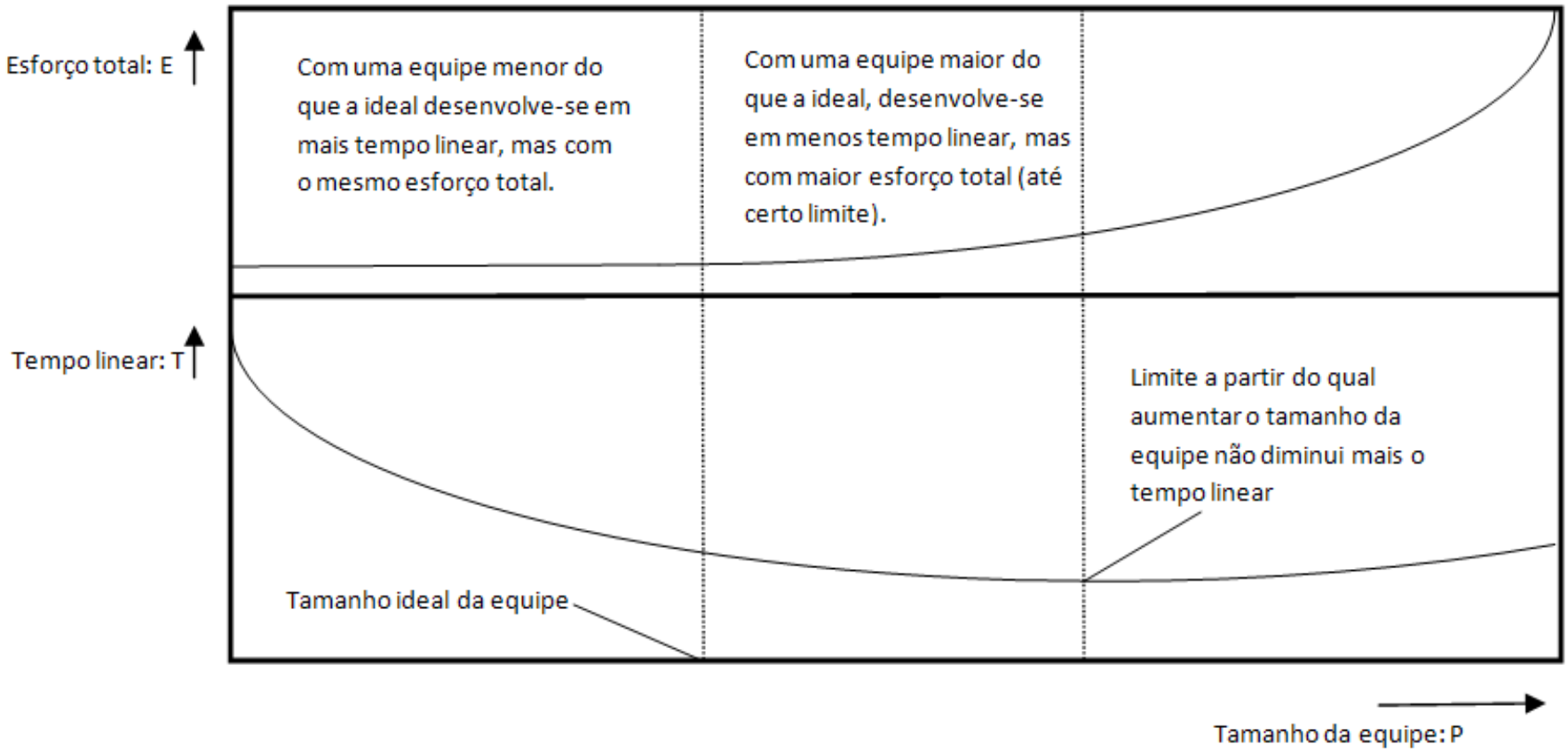
$$E = A * KSLOC^S * \prod_{i=1}^{n} M_i$$

- Durante as fases de Concepção e início da Elaboração usa-se o *Early Design Model* (Seção 0), com 6 multiplicadores (n=6).

- Mais tarde pode-se usar o *Post-Architecture Model*, com 16 multiplicadores (n=16).



Relação entre tamanho de equipe, esforço e tempo linear



Tempo linear

$$T = C * (E)^{D+0,2*(S-B)}$$

Onde:

a) T é o tempo linear ideal de desenvolvimento.
b) B, C e D são constantes que devem ser calibradas a partir de dados históricos (ver abaixo).
c) E é o esforço total para o projeto, conforme calculado anteriormente.
d) S é o expoente de esforço que já foi mencionado.

a) $A = 2,94$
b) $B = 0,91$
c) $C = 3,67$
d) $D = 0,28$

Fatores de escala

$$E = A * KSLOC^{S} * \prod_{i}^{n} M_i$$

$$S = B + 0{,}001 * \sum_{j=1}^{5} F_j$$

- Os cinco fatores de escala receberão cada um uma nota que varia de “muito baixo” até “extremamente alto”.

- Os fatores de escala terão impacto exponencial no tempo de desenvolvimento.

Fatores de escala

- *Precedentes* (*PREC*):
  - Se o produto é similar a vários projetos desenvolvidos anteriormente, então PREC é alto.
- *Flexibilidade no Desenvolvimento* (*FLEX*):
  - Se o produto deve ser desenvolvido estritamente dentro dos requisitos, é preso a definições de interfaces externas, então FLEX é baixo.
- *Arquitetura/Resolução de Riscos* (*RESL*):
  - Se existe bom suporte para resolver riscos e para definir a arquitetura então RESL é alto.
- *Coesão da Equipe* (*TEAM*):
  - Se a equipe é bem formada e coesa, então TEAM é alto.
- *Maturidade de Processo* (*PMAT*):
  - Este fator pode estar diretamente associado com o nível de maturidade CMMI. Quanto mais alto o nível de maturidade, maior será PMAT.

Precedentes

Tabela 7-7: Forma de obtenção do equivalente numérico para PREC.

Característica	Muito baixo Baixo	Nominal Alto	Muito alto Extra-alto
Compreensão organizacional dos objetivos do produto	Geral	Considerável	Total
Experiência no trabalho com sistemas de software relacionados	Moderada	Considerável	Extensiva
Desenvolvimento concorrente de novo hardware e procedimentos operacionais associados	Extensivo	Moderado	Algum
Necessidade de arquiteturas e algoritmos de processamento de dados inovadores	Considerável	Algum	Mínimo

Nota média	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Interpretação	Totalmente sem precedentes	Largamente sem precedentes	Um tanto sem precedentes	Genericamente familiar	Altamente familiar	Totalmente familiar
Fator numérico	6,20	4,96	3,72	2,48	1,24	0,00

Flexibilidade no desenvolvimento

Tabela 7-8: Forma de obtenção do equivalente numérico para FLEX.

Característica	Muito baixo Baixo	Nominal Alto	Muito alto Extra-alto
Necessidade de conformação do software a requisitos pré-estabelecidos	Total	Considerável	Básica
Necessidade de conformação do software a especificações de interfaces com sistemas externos	Total	Considerável	Básica
Combinação das inflexibilidades acima com prêmio por término antecipado do projeto	Alto	Médio	Baixo

Nota média	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Interpretação	Rigoroso	Relaxamento ocasional	Algum relaxamento	Conformidade geral	Alguma conformidade	Metas gerais
Fator numérico	5,07	4,05	3,04	2,03	1,01	0,00

Resolução de riscos

Tabela 7-9: Forma de obtenção do equivalente numérico para RESL.

Característica	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
O plano de gerenciamento de risco identifica todos os itens de risco críticos e estabelece marcos para resolvê-los	Nada	Um pouco	Alguma coisa	Geralmente	Largamente	Totalmente
Cronograma, orçamento e marcos internos são compatíveis com o plano de gerenciamento de risco	Nada	Um pouco	Alguma coisa	Geralmente	Largamente	Totalmente
Percentual do cronograma de desenvolvimento devotado a estabelecer a arquitetura, uma vez definidos os objetivos gerais do produto	5	10	17	25	33	40
Percentual de arquitetos de software experientes (*top*) disponíveis para o projeto em relação ao considerado necessário	20	40	60	80	100	120
Suporte de ferramentas disponível para resolver itens de risco, desenvolver e verificar especificações arquiteturais	Nenhum	Pouco	Algum	Bom	Forte	Total
Nível de incerteza nos determinantes-chave da arquitetura: missão, interface com usuário, COTS, hardware, tecnologia, desempenho	Extremo	Significativo	Considerável	Algum	Pouco	Muito pouco
Número de itens de risco e sua importância	Mais de 10 críticos	5 a 10 críticos	2 a 4 críticos	1 crítico	Mais de 5 não críticos	Menos de 5 não críticos

Resolução de riscos

Nota média	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Interpretação	Pouco (20%)	Algum (40%)	Frequente (60%)	Geralmente (75%)	Largamente (90%)	Totalmente (100%)
Fator numérico	7,07	5,65	4,24	2,83	1,41	0,00

Coesão da equipe de desenvolvimento

Tabela 7-10: Forma de obtenção do equivalente numérico para TEAM.

Característica	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Consistência dos objetivos e cultura dos interessados	Pouca	Alguma	Básica	Considerável	Forte	Total
Habilidade e vontade dos interessados em acomodar os objetivos de outros interessados	Pouca	Alguma	Básica	Considerável	Forte	Total
Experiência dos interessados em trabalhar como uma equipe	Nenhuma	Pouca	Pouca	Básica	Considerável	Extensiva
Construção de equipes com os interessados para obter visão compartilhada e compromissos	Nenhuma	Pouca	Pouca	Básica	Considerável	Extensiva

Nota média	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Interpretação	Interações muito difíceis	Algumas interações difíceis	Interações basicamente cooperativas	Predominantemente cooperativas	Altamente cooperativas	Interações perfeitas
Fator numérico	5,48	4,38	3,29	2,19	1,10	0,00

Maturidade do processo

Tabela 7-11: Forma de obtenção do equivalente numérico para PMAT.

Característica	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Nível CMM (ou CMMI)	1 – inferior	1 – superior	2	3	4	5
Nível EPML (ou SPICE)	0	1	2	3	4	5

Nota média	Muito baixo	Baixo	Nominal	Alto	Muito alto	Extra-alto
Interpretação	Sem processo definido	Processo incipiente	Processo definido	Processo gerenciado	Processo padronizado gerenciado quantitativamente	Processo em otimização constante
Fator numérico	7,80	6,24	4,68	3,12	1,56	0,00

Multiplicadores de esforço

$$E = A * KSLOC^S * \prod_{i}^{n} M_i$$

$$S = B + 0{,}001 * \sum_{j=1}^{5} F_j$$

- Os multiplicadores de esforço são usados para ajustar a estimativa de esforço para o desenvolvimento de um sistema baseando-se em características próprias do projeto e da equipe que podem onerar este tempo.

Multiplicadores de Esforço do Post-Architecture Model

- Fatores do Produto.
- Fatores da Plataforma.
- Fatores Humanos.
- Fatores de Projeto.

Fatores do produto

- *Software com Confiabilidade Requerida* (RELY).
- *Tamanho da Base de Dados* (DATA).
- *Complexidade do Produto* (CPLX).
- *Desenvolvimento Visando Reusabilidade* (RUSE).
- *Documentação Necessária para o Ciclo de Desenvolvimento* (DOCU).

Confiabilidade requerida

Tabela 7-12: Forma de obtenção do equivalente numérico para RELY.

Descritor	Pequena inconveniência	Perdas pequenas, facilmente recuperáveis	Perdas moderadas, facilmente recuperáveis	Alta perda financeira	Risco a vida humana	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	0,82	0,92	1,00	1,10	1,26	n/a

Tamanho da base de dados

O multiplicador de esforço *DATA* (*Tamanho da Base de Dados*) avalia o tamanho relativo da base de dados usada *para testes* do programa (não a base de dados final). A razão D/P é o número de *Kbytes* na base de dados de teste (*D*) dividido pelo número de milhares de linhas (*P*) estimado do programa (em KSLOC). A Tabela 7-13 apresenta os parâmetros de cálculo para DATA.

Tabela 7-13: Forma de obtenção do equivalente numérico para DATA.

Descritor		D/P < 10	10 ≤ D/P ≤ 100	100 ≤ D/P ≤ 1000	DP ≥ 1000	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	n/a	0,90	1,00	1,14	1,28	n/a

Complexidade do produto 1/2

Tabela 7-14: Forma de obtenção do equivalente numérico para CPLX.

Operações de controle	Código sequencial com poucas estruturas não aninhadas. Composição simples de módulos via chamada de procedimentos ou *scripts*	Aninhamento simples de estruturas de controle. Basicamente predicados simples.	Basicamente aninhamento simples. Algum controle intermódulos. Tabelas de decisão. Chamadas ou passagem de mensagens, incluindo processamento distribuído suportado por middleware.	Estruturas altamente aninhadas com vários predicados compostos. Controle de fila e pilha. Processamento distribuído homogêneo. Controle de tempo real simples em processador único.	Código reentrante e recursivo. Gerenciamento de interrupção com prioridade fixa. Sincronização de tarefas. Chamadas complexas. Processamento distribuído heterogêneo. Controle de tempo real complexo em processador único.	Escalonamento de múltiplos recursos com mudança dinâmica de prioridades. Controle em nível de micro código. Controle complexo de tempo real distribuído.
Operações computacionais	Avaliação de expressões simples como A:=B+C*(D-E)	Avaliação de expressões de nível moderado como D:=SQRT(B**2-4.*A*C)	Uso de rotinas matemáticas e estatísticas padrão. Operações básicas sobre matrizes e vetores.	Análise numérica básica: interpolação multivariada e equações diferenciais ordinárias. Arredondamento e truncamento básicos.	Análise numérica complexa, mas estruturada: equações de matrizes, equações diferenciais parciais. Paralelização simples.	Análise numérica complexa e não estruturada: análise de ruído altamente precisa, dados estocásticos. Paralelização complexa.

Complexidade do produto 2/2

Operações dependentes de dispositivo	Comandos simples de leitura e escrita com formatação simples.	Sem necessidade de conhecimento de características particulares de processador ou dispositivo de E/S. E/S feita por Get e Put.	Processamento de E/S inclui seleção de dispositivo, checagem de *status* e processamento de erros.	Operações de E/S em nível físico (traduções de endereços de armazenamento físicos; buscas e leituras, etc.). Overlap de E/S otimizado.	Rotinas para diagnóstico de interrupção. Gerenciamento de linha de comunicação. Sistemas embarcados com consideração intensiva de performance.	Codificação de dispositivos dependentes de tempo. Operações microprogramadas. Sistemas embutidos com performance crítica.
Operações de gerenciamento de dados	Arrays simples em memória. Simples consultas e atualizações em COTS ou banco de dados.	Arquivos simples sem edição nem buffers. Consultas e atualizações em bancos de dados ou COTS moderadamente complexas.	Entrada de múltiplos arquivos e saída em arquivo único. Mudanças estruturais simples. Edição simples. Consultas e atualizações complexas em COTS ou banco de dados.	Gatilhos simples ativados pelo conteúdo de sequências de dados. Reestruturação de dados complexa.	Coordenação de bancos de dados distribuídos. Gatilhos complexos. Otimização.	Estruturas relacionais e de objetos dinâmicas e altamente acopladas. Gerenciamento de dados em linguagem natural.
Operações de gerenciamento de interface com usuário	Formulários de entrada simples e geradores de relatórios.	Uso de construtores de interface com usuário (GUI) simples	Simples uso de um conjunto de *widgets*.	Desenvolvimento e extensão de conjunto de widgets. E/S por voz. Multimídia.	Gráficos dinâmicos 2D e 3D moderadamente complexos. Multimídia.	Multimídia complexa. Realidade virtual. Interface em linguagem natural.
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	0,73	0,87	1,00	1,17	1,34	1,74

Desenvolvimento visando reusabilidade

Tabela 7-15: Forma de obtenção do equivalente numérico para RUSE.

Descritor		Nenhum reuso	Dentro do projeto	Dentro de um programa	Dentro de uma *SPL*	Entre múltiplas *SPLs*
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	n/a	0,95	1,00	1,07	1,15	1,24

Documentação necessária para o ciclo de desenvolvimento

Tabela 7-16: Forma de obtenção do equivalente numérico para DOCU.

Descritor	Muitas necessidades de ciclo de vida não cobertas	Algumas necessidades de ciclo de vida não cobertas	Exatamente dimensionada para as necessidades do ciclo de vida	Excessiva para as necessidades do ciclo de vida	Muito excessiva para as necessidades do ciclo de vida	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	0,81	0,91	1,00	1,11	1,23	n/a

Multiplicadores de esforço referentes à plataforma

- *Restrição de Tempo de Execução* (TIME).
- *Restrição de Memória Principal* (STOR).
- *Volatilidade da Plataforma* (PVOL).

Restrição de tempo de execução

Tabela 7-17: Forma de obtenção do equivalente numérico para TIME.

Descritor			Menos de 50% de uso do tempo de execução disponível	70% de uso do tempo de execução disponível	85% de uso do tempo de execução disponível	95% de uso do tempo de execução disponível.
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	n/a	n/a	1,00	1,11	1,29	1,63

Restrição de memória principal

Tabela 7-18: Forma de obtenção do equivalente numérico para STOR.

Descritor			Menos de 50% de uso da memória principal	70% de uso da memória principal	85% de uso da memória principal	95% de uso da memória principal
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	n/a	n/a	1,00	1,05	1,17	1,46

Volatilidade da plataforma

Tabela 7-19: Forma de obtenção do equivalente numérico para PVOL.

Descritor		Mudanças grandes a cada 12 meses, pequenas a cada mês	Grandes: 6 meses; pequenas: 2 semanas	Grandes: 2 meses; pequenas: 1 semana	Grandes: 2 semanas; pequenas: 2 dias	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	n/a	0,87	1,00	1,15	1,30	n/a

Fatores humanos

- *Capacidade dos Analistas* (ACAP).
- *Capacidade dos Programadores* (PCAP).
- *Continuidade de Pessoal* (PCON).
- *Experiência em Aplicações Semelhantes* (APEX).
- *Experiência na Plataforma* (PLEX).
- *Experiência na Linguagem e Ferramentas* (LTEX).

Capacidade dos analistas

Tabela 7-20: Forma de obtenção do equivalente numérico para ACAP.

Descritor	Percentil 15	35	55	75	90	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,42	1,19	1,00	0,85	0,71	n/a

Capacidade dos programadores

Tabela 7-21: Forma de obtenção do equivalente numérico para PCAP.

Descritor	Percentil 15	35	55	75	90	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,34	1,15	1,00	0,88	0,76	n/a

Continuidade de pessoal

Tabela 7-22: Forma de obtenção do equivalente numérico para PCON.

Descritor	48%/ano	24%/ano	12%/ano	6%/ano	3%/ano	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,29	1,12	1,00	0,90	0,81	n/a

Experiência em aplicações semelhantes

Tabela 7-23: Forma de obtenção do equivalente numérico para APEX.

Descritor	Menos de 2 meses	6 meses	1 ano	3 anos	6 anos	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,22	1,10	1,00	0,88	0,81	n/a

Experiência na plataforma

Tabela 7-24: Forma de obtenção do equivalente numérico para PLEX.

Descritor	Menos de 2 meses	6 meses	1 ano	3 anos	6 anos	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,19	1,09	1,00	0,91	0,85	n/a

Experiência na linguagem e ferramentas

Tabela 7-25: Forma de obtenção do equivalente numérico para LTEX.

Descritor	Menos de 2 meses	6 meses	1 ano	3 anos	6 anos	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,20	1,09	1,00	0,91	0,84	n/a

Fatores de projeto

Uso de Ferramentas de Software (TOOL).
Equipe de Desenvolvimento Distribuída (SITE).
Cronograma de Desenvolvimento Requerido (SCED).

Uso de ferramentas de software

Tabela 7-26: Forma de obtenção do equivalente numérico para TOOL.

Descritor	Editar, codificar, debugar.	CASE simples. Pouca integração.	Ferramentas básicas de ciclo de vida moderadamente integradas.	Ferramentas de ciclo de vida fortes e maduras, moderadamente integradas	Ferramentas de ciclo de vida fortes, maduras e bem integradas com processos, métodos e reuso	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,17	1,09	1,00	0,90	0,78	n/a

Equipe de desenvolvimento distribuída

Tabela 7-27: Forma de obtenção do equivalente numérico para SITE.

Descritor de co-locação	Internacional	Multi-cidade e multi-empresa	Multi-cidade ou multi-empresa	Mesma cidade ou área metropolitana	Mesmo edifício ou complexo	Totalmente co-locada
Descritor de comunicação	Alguns telefones, correio	Telefones individuais, FAX	Email	Comunicação eletrônica de banda larga	Videoconferência	Multimídia interativa
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,22	1,09	1,00	0,93	0,86	0,80

Cronograma de desenvolvimento requerido

Tabela 7-28: Forma de obtenção do equivalente numérico para SCED.

Descritor	75% do tempo nominal	85%	100%	130%	160%	
Avaliação	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra-alto
Equivalente numérico	1,43	1,14	1,00	1,00	1,00	n/a

Fatores do Early design model

- *Capacidade de Pessoal* (PERS).
- *Confiabilidade e Complexidade do Produto* (RCPX).
- *Desenvolvimento para Reuso* (RUSE).
- *Dificuldade com a Plataforma* (PDIF).
- *Experiência do Pessoal* (PREX).
- *Instalações* (FCIL).
- Cronograma de Desenvolvimento Requerido (SCED).

Esforço e tempo por fase RUP

Fase	Esforço nominal	Intervalo	Tempo linear nominal	Intervalo
Concepção	0,06*E*	0,02 a 0,15	0,125*T*	0,02 a 0,3
Elaboração	0,24*E*	0,20 a 0,28	0,375*T*	0,33 a 0,42
Construção	0,76*E*	0,72 a 0,80	0,625*T*	0,58 a 0,67
Transição	0,12*E*	0,00 a 0,20	0,125*T*	0,00 a 0,20
Totais	**1,18*E***		**1,25*T***	

Exemplo

Assim, por exemplo, um projeto com E = 56 desenvolvedor-mês e T = 11,5 meses terá, em média os valores de esforço (em desenvolvedor-mês) e duração por fase definidos como na Tabela 7-35.

Tabela 7-35: Exemplo de cálculo de tempo e esforço para as fases do UP de um projeto com E = 56 e T = 11,5.

Fase	Esforço (desenvolvedor-mês)	Tempo (meses)
Concepção	3,4	1,4
Elaboração	13,4	4,3
Construção	42,6	7,2
Transição	6,7	1,4
Totais	**66,1**	**14,3**

Esforço por disciplina

Tabela 7-36: Resumo do esforço relativo às disciplinas UP nas diferentes fases.

Disciplina	Concepção	Elaboração	Construção	Transição
Gerenciamento	14%	12%	10%	14%
Ambiente/Configuração	10%	8%	5%	5%
Requisitos	38%	18%	8%	4%
Design	19%	36%	16%	4%
Implementação	8%	13%	34%	19%
Avaliação/Teste	8%	10%	24%	24%
Implantação	3%	3%	3%	30%
Total	**100%**	**100%**	**100%**	**100%**

Calibragem do modelo

$$E = A * KSLOC^{S} * \prod_{i=1}^{n} M_i$$

Tendo realizado pelo menos 5 projetos...

Tabela 7-37: Exemplo de calibragem para a constante *A*.

Real	$KSLOC^S * \prod_{i=1}^{n} M_i$	*ln(Real)*	$ln(KSLOC^S * \prod_{i=1}^{n} M_i)$	$ln(Real)-ln(KSLOC^S * \prod_{i=1}^{n} M_i)$
1854,6	686,7	7,53	6,53	0,99
258,5	94,3	5,55	4,55	1,01
201,0	77,7	5,30	4,35	0,95
58,9	20,3	4,08	3,01	1,07
9661,0	3338,8	9,18	8,11	1,06
7021,3	2753,5	8,86	7,92	0,94
91,7	38,9	4,52	3,66	0,86
689,7	301,1	6,54	5,71	0,83
				X = 0,96
				A = 2,62

$$A = e^X$$

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*.

Esforço de um Projeto

- Uma vez que o *AFP* do projeto tenha sido calculado, o esforço total será calculado multiplicando-se o *AFP* pelo índice de produtividade (*IP*) da equipe.

- Esse índice deve ser calculado para o ambiente local, e pode variar muito em função do ambiente de trabalho, experiência da equipe e outros fatores.

- Assim, o esforço total do projeto é calculado como:

$$E = AFP * IP$$

Custo do projeto

$Custo = E * Custo_{hora}$

- www.fattocs.com.br/editais.asp
  - armazena editais brasileiros de contratação de software onde a medida de custo é o ponto de função.
  - No site, o preço por ponto de função varia de 100 a 1000 reais, a maioria ficando entre 400 e 600 reais, o que pode ser explicado pelo tipo de sistema que se está contratando.

Tempo linear

- Pode-se usar a fórmula de COCOMO II ou:

$$T = 2{,}5 * \sqrt[3]{E}$$

$$P = E/T$$

- Tempo mínimo (o esforço será maior que *E*):

$$T_{min} = 0{,}75 * \sqrt[3]{E}$$

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.

Comunicação de dados

- Avalia o grau em que necessidades especiais de comunicação afetam o sistema:
  - 0: para aplicações que são somente processamento em *batch* ou que rodam isoladas em um PC.
  - 1: para aplicações em *batch* mas com entrada de dados remota *ou* saída remota.
  - 2: para aplicações em *batch* com entrada de dados remota *e* saída remota.
  - 3: para aplicações que incluem coleta de dados *online* ou *front-end* de teleprocessamento para um sistema em *batch* ou sistema de consultas.
  - 4: para aplicações que são mais do que um *front-end*, mas suportam um único tipo de protocolo de comunicação.
  - 5: para aplicações que são mais do que um *front-ent* e suportam vários tipos de protocolos de comunicação.

Processamento de dados distribuído

- Avalia o grau em que dados distribuídos são usados pela aplicação:
  - 0: para aplicações que não ajudam na transferência de dados ou funções de processamento entre os componentes do sistema.
  - 1: para aplicações que preparam os dados para o processamento do usuário final em outro componente do sistema, tal como sistemas que geram dados para serem lidos em uma planilha ou arquivo de processador de texto.
  - 2: para aplicações que preparam dados para transferência e então transferem e processam os dados em outro componente do sistema (não para processamento do usuário final).
  - 3: para aplicações onde o processamento distribuído e transferência de dados ocorrem *online* e em uma direção apenas.
  - 4: para aplicações onde o processamento distribuído e transferência de dados ocorrem *online* e nas duas direções.
  - 5: para aplicações onde as funções são executadas dinamicamente no componente mais apropriado do sistema.

Performance

- Avalia o grau em que a eficiência do sistema precisa ser considerada em sua construção. Sistemas eficientes sempre são desejáveis, mas este fator avalia o quanto a eficiência é critica para o sistema, de forma que se invistam recursos de tempo e dinheiro para melhorar esse aspecto.
  - 0: nenhum requisito de performance especial foi definido pelo cliente.
  - 1: requisitos de performance foram estabelecidos e revisados, mas nenhuma ação especial precisa ser tomada.
  - 2: tempo de resposta e taxa de transferência são críticos durante as horas de pico. Nenhum *design* especial para utilização de CPU é necessário. O prazo para a maioria dos processamentos é o dia seguinte.
  - 3: o tempo de resposta e taxa de transferência são críticos durante o horário comercial. Nenhum *design* especial para utilização de CPU é necessário. Os requisitos de prazo de processamento com sistemas interfaceados são restritivos.
  - 4: em adição, os requisitos de performance são suficientemente restritivos para que se necessite estabelecer tarefas de análise de performance durante a fase de *design*.
  - 5: em adição, ferramentas de análise de performance devem ser usadas nas fases de *design*, desenvolvimento e/ou implementação para atender os requisitos de performance do cliente.

Uso do sistema

- Avalia o grau em que o sistema necessita ser projetado para compartilhar recursos de processamento:
  - 0: nenhuma restrição operacional implícita ou explícita é incluída.
  - 1: restrições operacionais existem mas são menos restritivas do que em uma aplicação típica. Nenhum esforço especial é necessário para satisfazer as restrições.
  - 2: são incluídas algumas considerações sobre tempo e segurança.
  - 3: requisitos específicos de processador para uma parte específica da aplicação são incluídos.
  - 4: restrições sobre operações estabelecidas requerem que a aplicação tenha um processador dedicado ou prioridade de tempo no processador central.
  - 5: em adição, existem restrições especiais na aplicação em relação aos componentes distribuídos do sistema.

Taxa de transações

- Avalia a quantidade de transações simultâneas esperada:
  - 0: não são antecipados períodos de picos de transações.
  - 1: períodos de picos de transações (por exemplo, mensalmente, semestralmente, anualmente) são antecipados.
  - 2: picos de transação semanais são antecipados.
  - 3: picos de transação diários são antecipados.
  - 4: altas taxas de transação são estabelecidas pelo cliente nos requisitos da aplicação ou nos acordos de nível de serviço, as quais são suficientemente altas para necessitar de atividades de análise de performance na fase de *design*.
  - 5: em adição, se requer o uso de ferramentas de análise de performance nas fases de *design*, desenvolvimento e/ou instalação.

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.

Eficiência do usuário final

- Avalia o grau em que a aplicação será projetada para melhorar a eficiência do usuário final.
  - 0: nenhuma das opções abaixo.
  - 1: de uma a três das opções abaixo.
  - 2: de quatro a cinco das opções abaixo.
  - 3: seis ou mais das opções abaixo, mas não há requisitos específicos relacionados a eficiência de usuário final.
  - 4: seis ou mais das opções abaixo, e requisitos estabelecidos para a eficiência de usuário final são suficientemente fortes para requerer a inclusão de atividades de *design* para fatores humanos (por exemplo minimizar a quantidade de *clicks* e movimentos de mouse, maximização de *defaults* e uso de *templates*).
  - 5: seis ou mais das opções abaixo, e os requisitos estabelecidos para a eficiência de usuário são suficientemente fortes para requerer o uso de ferramentas e processos especiais para demonstrar que os objetivos foram atingidos.
    - Ajuda navegacional (por exemplo, teclas de função, menus gerados dinamicamente, etc.).
    - Menus.
    - Ajuda e documentação *online*.
    - Movimentação de cursor automatizada.
    - *Scrolling*.
    - Impressão remota (a partir de transações *online*).
    - Teclas de função pré-definidas.
    - Tarefas em *batch* submetidas a partir de transações *online*.
    - Seleção por cursor na tela de dados.
    - Alto uso de cores e destaque visual em tela.
    - Copias impressas de documentação de usuário de transações *online*.
    - Interface por mouse.
    - Janelas *pop-up*.
    - Minimização do número de janelas para realizar objetivos de negócio.
    - Suporte bilíngue (conta como quatro itens).
    - Suporte multilíngue (conta como seis itens).

Atualização online

- Avalia o percentual de arquivos internos que podem ser atualizados de forma *online*:
  - 0: nenhuma atualização *online*.
  - 1: é incluída a atualização *online* para um a três arquivos. O volume de atualização é baixo e a recuperação é simples.
  - 2: a atualização *online* de quatro ou mais arquivos é incluída. O volume de atualização é baixo e a recuperação é simples.
  - 3: a atualização *online* dos principais arquivos lógicos internos é incluída.
  - 4: em adição, proteção contra a perda de dados é essencial, e o sistema deve ser especialmente projetado contra perda de dados.
  - 5: em adição, altos volumes de atualização trazem considerações de custo para o processo de recuperação. Procedimentos de recuperação altamente automatizados com intervenção mínima do operador são incluídos.

Processamento complexo

- Avalia o grau em que a aplicação utiliza processamento lógico ou matemático complexo.
  - 0: nenhuma das opções abaixo.
  - 1: qualquer uma das opções abaixo.
  - 2: quaisquer duas das opções abaixo.
  - 3: quaisquer três das opções abaixo.
  - 4: quaisquer quatro das opções baixo.
  - 5: todas as cinco opções abaixo.
    - Controle cuidadoso (por exemplo, processamento especial de auditoria) e/ou processamento seguro específico da aplicação.
    - Processamento lógico extensivo.
    - Processamento matemático extensivo.
    - Muito processamento de exceções resultante de transações incompletas que precisam ser processadas novamente, como, por exemplo, transações de caixa-automático incompletas causadas por interrupção de teleprocessamento, valores de dados que faltam ou edições que falharam.
    - Processamento complexo para gerenciar múltiplas possibilidades de entrada e saída, como, por exemplo, multimídia ou independência de dispositivos.

Reusabilidade

- Avalia em que grau a aplicação é projetada para ser reusável:
  - 0: não há nenhuma preocupação para produzir código reusável.
  - 1: código reusável é gerado para uso dentro da própria aplicação.
  - 2: menos de 10% da aplicação deve considerar mais do que simplesmente as necessidades do usuário.
  - 3: 10% ou mais da aplicação deve considerar mais do que as necessidades do usuário.
  - 4: a aplicação deve ser especificamente empacotada e/ou documentada para facilitar o reuso, e a aplicação deve ser personalizável pelo usuário em nível de código fonte.
  - 5: a aplicação deve ser especificamente empacotada e/ou documentada para facilitar o reuso, e a aplicação deve ser personalizável por meio de manutenção de usuário baseada em parâmetros.

Facilidade de instalação

- Avalia em que grau haverá preocupação em tornar fácil a instalação do sistema e a conversão dos dados.
  - 0: nenhuma consideração especial foi estabelecida pelo usuário, e nenhum *setup* especial é necessário para a instalação.
  - 1: nenhuma consideração especial foi estabelecida pelo usuário, mas um *setup* especial é requerido para instalação.
  - 2: requisitos de conversão e instalação foram estabelecidos pelo usuário, e guias de conversão e instalação devem ser fornecidas e testadas. O impacto da conversão no projeto não é considerado importante.
  - 3: requisitos de conversão e instalação de foram estabelecidos pelo usuário, e guias de conversão e instalação devem ser fornecidas e testadas. O impacto da conversão no projeto é considerado importante.
  - 4: em adição à nota 2 acima, ferramentas de conversão e instalação automática devem ser fornecidas e testadas.
  - 5: em adição à nota 3 acima, ferramentas de conversão e instalação automática devem ser fornecidas e testadas.

Facilidade de operação

- Avalia em que grau a aplicação:
  - 0: nenhuma consideração operacional especial além dos procedimentos normais de *backup* foram estabelecidos pelo usuário.
  - 1-4: um, alguns ou todos os itens abaixo se aplicam ao sistema (Deve-se selecionar todos os que se aplicam. Cada item vale um ponto, exceto se for dito o contrário):
    - Processos efetivos de inicialização, *backup* e recuperação devem ser fornecidos, mas a intervenção do operador é necessária.
    - Processos efetivos de inicialização, *backup* e recuperação devem ser fornecidos, e nenhuma intervenção do operador é necessária (conta como dois itens).
    - A aplicação deve minimizar a necessidade de armazenamento em fitas (ou qualquer outro meio de armazenamento *offline*).
    - A aplicação deve minimizar a necessidade de manuseio de papel.
  - 5: a aplicação é projetada para operar de forma não supervisionada. Não supervisionada significa que não é necessária nenhuma intervenção do operador do sistema a não ser, talvez, na sua primeira inicialização ou desligamento final. Uma das características da aplicação é a recuperação automática de erros.

Múltiplos locais

- Avalia o grau em que a aplicação é projetada para funcionar de forma distribuída.
  - 0: requisitos do usuário não exigem a consideração de necessidade de mais do que um usuário ou instalação.
  - 1: a necessidade de múltiplos locais deve ser considerada no projeto, e a aplicação deve ser projetada para operar apenas em ambientes idênticos de hardware e software.
  - 2: a necessidade de múltiplos locais deve ser considerada no projeto, e aplicação deve ser projetada para operar apenas em ambientes de hardware e software similares.
  - 3: a necessidade de múltiplos locais deve ser considerada no projeto e aplicação é projetada para operar em ambientes de hardware e software diferentes.
  - 4: o plano de documentação e suporte deve ser fornecido e testado para suportar a aplicação em múltiplos locais, e a aplicação é como descrita nas notas um ou dois.
  - 5: o plano de documentação e suporte deve ser fornecido e testado para suportar a aplicação em múltiplos locais, e a aplicação é como descrita na nota três.

Facilidade para mudança

- Avalia o grau em que a aplicação é projetada para facilitar mudanças lógicas e estruturais.
  - 0: nenhum item abaixo.
  - 1: um item.
  - 2: dois itens.
  - 3: três itens.
  - 4: quatro itens.
  - 5: cinco itens ou mais.
- Facilidades de consulta e relatório flexíveis devem ser fornecidas para tratar consultas simples, por exemplo, operadores lógicos binários aplicados apenas a um arquivo lógico interno (conta como um item).
- Facilidades de consulta e relatório flexíveis devem ser fornecidas para tratar consultas de complexidade média, por exemplo, operadores lógicos binários aplicados a mais do que um arquivo lógico interno (conta como dois itens).
- Facilidades de consulta e relatório flexíveis devem ser fornecidas para tratar consultas de complexidade alta, por exemplo, combinações de operadores lógicos binários em um ou mais arquivos lógicos internos (conta como três itens).
- Dados de controle de negócio são mantidos em tabelas que são gerenciadas pelo usuário e com processos interativos *online*, mas as mudanças só têm efeito no dia seguinte (contra como um item).
- Dados de controle de negócio são mantidos em tabelas que são gerenciadas pelo usuário e com processos interativos *online*, e as mudanças têm feito imediatamente (conta como dois itens).

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

$$UCP = UUCP * TCF * EF$$

Esforço

$$E = UCP * IP$$

Pontos de História (*PH*)

- É a estimativa de esforço preferida (embora não exclusiva) de métodos ágeis como *Scrum* e XP.

- Um ponto de história, não é uma medida de complexidade funcional como pontos de função ou pontos de caso de uso, mas uma medida de esforço relativa à equipe de desenvolvimento.

- Segundo Kniberg (2007) uma estimativa baseada em pontos de histórias deve ser feita pela equipe.

- Inicialmente pergunta-se à equipe quanto tempo tantas pessoas que se dedicassem unicamente a uma história de usuário levariam para terminá-la, gerando uma versão executável funcional.

- Se a resposta for, por exemplo, “3 pessoas levariam 4 dias”, então atribua à história 3x4 = 12 pontos de história.

- Assim, um ponto de história pode ser definido como o esforço de desenvolvimento de uma pessoa durante um dia ideal de trabalho, lembrando que o *dia ideal* de trabalho consiste em uma pessoa dedicada durante 6 a 8 horas a um projeto, sem interrupções nem atividades paralelas.

Atribuição de Pontos de História

- Nos métodos ágeis, a importância da estimativa normalmente está na comparação entre histórias, ou seja, mais importante do que saber quantos dias uma história efetivamente levaria para ser implementada é saber que uma história levaria duas vezes mais tempo do que outra para ser implementada.

Fibonacci

- Os pontos de história são atribuídos normalmente não como valores da série dos números naturais, mas como valores da série aproximada de números de Fibonacci.
- Um número de Fibonacci é definido como a soma dos dois números de Fibonacci anteriores na série (com exceção dos dois primeiros, que por definição são 1 e 1).
- Assim, o início da série de Fibonacci é constituído pelos números: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, etc.
- Porém, pode ser estranho mensurar pontos de história em 89 ou 34 pontos.
- Então na prática acaba-se fazendo uma aproximação desses valores para uma série como: 1, 2, 3, 5, 8, 15, 25, 40, 60, 100, etc.
- A ideia é que os pontos de história apresentem uma ordem de grandeza natural para o esforço e não uma medida exata.

Camiseta

- Outra opção para estimar pontos de história é usar o sistema “camiseta”, com valores “pequeno”, “médio” e “grande”.

O procedimento de atribuição de pontos funciona assim:

- Toma-se da lista de histórias de usuário previamente preparada aquelas consideradas mais simples, e atribui-se a elas 1 ou 2 pontos.
-
- Depois, sequencialmente, vai-se pegando outras mais complexas, inicialmente de 3, depois de 5 pontos e assim por diante.

- Segundo Toledo (2009), o motivo é que, para o ser humano é muito mais fácil fazer medidas relativas do que absolutas.

- É difícil uma pessoa estimar o peso de um cavalo, sem ter uma balança ou conhecimento prévio do valor.

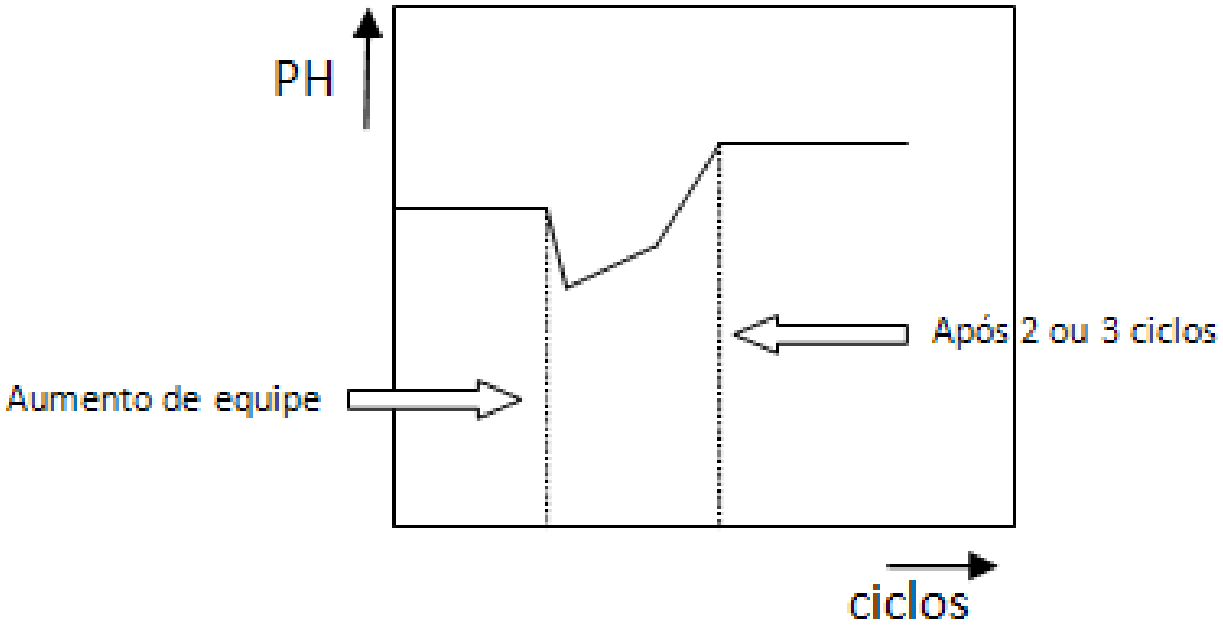
- Mas uma pessoa consegue estimar facilmente que um cavalo pesa menos do que um elefante e mais do que um cachorro.

- Felix (2009) comenta que a atribuição de pontos de história usualmente segue critérios subjetivos de complexidade, esforço e risco, sendo caracterizada por frases como, por exemplo:
  - *Complexidade*: “Essa regra de negócio tem muitos cenários possíveis”.
  - *Esforço*: “Essa alteração é simples, mas precisa ser realizada em muitas telas”.
  - *Risco*: “Precisamos utilizar o *framework* X, mas ninguém na equipe tem experiência”.

Medição de Velocidade

- Pontos de histórias são usados por equipes ágeis para medir sua velocidade de projeto.
- Pode-se fazer um gráfico e deixar a vista de todos onde a cada ciclo são medidos os pontos de história efetivamente desenvolvidos.
- Se esta velocidade começar a cair, a equipe deve verificar o motivo.
- Vários motivos podem ser listados:
  - desmotivação,
  - erros de estimação,
  - erros de priorização (a equipe começou a tratar histórias de usuário mais simples e deixou as mais complexas e arriscadas para depois), etc.

- Exemplo de gráfico de velocidade de projeto onde após a aquisição de novos membros para a equipe há uma redução na velocidade em PH que é compensada após 3 ciclos, quando então os novos membros da equipe passam a ser efetivamente produtivos.



- Normalmente um gráfico de velocidade é relativamente estável, embora os valores possam variar de um ciclo para outro, sua derivada se mantém constante.

- Se houver medidas de melhoria de produtividade pode-se esperar aumentos na velocidade, ou seja, uma derivada positiva.

- Então, a principal utilidade do gráfico de velocidade consiste em ajudar a diagnosticar possíveis problemas de ambiente, caso a derivada se torne negativa.