

## Capítulo 3

### Análise de Pontos por Função

---

*O objetivo deste capítulo é apresentar os conceitos da FPA, além de detalhar a forma de aplicá-los. Em seguida, os pontos positivos e negativos dessa técnica serão expostos, juntamente com algumas de suas principais extensões.*

Nos princípios da década de setenta, os pesquisadores da IBM iniciaram estudos para determinar que variáveis críticas poderiam determinar a produtividade na programação. Descobriu-se que um sistema seria melhor avaliado através das funções executadas pelos programas, em vez de considerar o volume ou a complexidade do código.

Em 1979, Allan J. Albrecht (Albrecht, 1979) definiu a Análise de Pontos por Função (*Function Point Analysis* - FPA) a partir desses estudos, buscando mapear as questões pertinentes à estimativa e avaliação de produtividade no desenvolvimento de software em ambientes heterogêneos. Segundo Smith (1997), as primeiras definições da FPA foram refinadas e estendidas no *IBM CIS Guideline 313, AD/M Productivity Measurement and Estimate Validation*, em 1984. Em 1986, um grupo de usuários da FPA formou o *International Function Point User Group* (IFPUG), o qual é responsável por manter seus associados informados a respeito das novas atualizações da técnica.

Segundo Pressman (2000), a FPA é uma métrica orientada a função, derivada a partir de medidas diretas, que dimensiona o software considerando a funcionalidade entregue ao usuário final. Resumidamente, o projeto lógico deve ser inteiramente decomposto em funções de acordo com os arquivos de dados, interfaces, entradas, saídas e consultas, atribuindo pesos a cada uma dessas funções. Esses pesos são multiplicados pelas quantidades de cada função e posteriormente somados. Finalmente, o somatório é ajustado conforme uma análise das características gerais de complexidade do sistema.

Conforme Tavares (1999), Ho (1999) e IFPUG (1999) os objetivos da FPA são:

- *Medir o que foi requisitado e recebido pelo usuário;*
- *Prover uma métrica para a qualidade e análise de produtividade;*
- *Prover um método para estimar o tamanho do software, independentemente da tecnologia utilizada para implementação e sob a perspectiva do usuário; e*
- *Prover um fator de normalização para comparação de software.*

O IFPUG tem demonstrado duas preocupações fundamentais em relação ao processo de contagem dos pontos (IFPUG, 1999):

1. *Manter sua simplicidade para evitar acréscimo excessivo de trabalho; e*
2. *Sustentar sua concisão para garantir a consistência, ao longo do tempo, dos projetos, e entre os usuários da técnica.*

Embora a idéia original tenha sido prover uma forma para medir tamanho de projetos de software, a FPA tem sido utilizada em outras atividades, tais como (Heller, 1995; IFPUG, 1999; Tavares, 1999):

- *Fornecer suporte para planejamento e controle de projetos de software;*
- *Apoiar a análise de produtividade e qualidade do processo de desenvolvimento de sistemas;*
- *Apoiar na gerência de mudança de escopo do projeto;*
- *Entender melhor o projeto;*
- *Servir-se como ferramenta de comunicação com os usuários;*
- *Estimar custos de desenvolvimento;*
- *Realizar estudos de benchmark;*
- *Facilitar a negociação nos contratos de software; e*
- *Medir os artefatos de um sistema (documentos, código fonte, materiais de teste, entre outros).*

A seção seguinte apresenta uma visão geral da FPA.

### 3.1. Visão Geral da FPA

A FPA pode ser aplicada para calcular o tamanho de projetos de desenvolvimento, de projetos de manutenção, ou de aplicações existentes, sendo este o procedimento inicial a ser executado para se determinar o tipo da contagem:

1. *Cálculo de um Projeto de Desenvolvimento*: calcula-se o tamanho de um projeto de desenvolvimento de uma nova aplicação, considerando as funções solicitadas e entregues ao usuário, inclusive aquelas utilizadas somente na implantação do projeto, chamadas de função de conversão de dados.
2. *Cálculo de um Projeto de Manutenção*: utilizado para dimensionar um projeto de manutenção de uma aplicação existente. Todas as inclusões, alterações e exclusões de funções do usuário e do processo de conversão de dados devem ser contabilizadas.
3. *Cálculo de uma Aplicação*: calcula-se o tamanho de uma aplicação existente, o que representa a funcionalidade da aplicação que está disponível para o usuário, sob o seu ponto de vista. Neste caso, as funções de conversão de dados não são incluídas.

O passo seguinte é a identificação do *escopo de contagem* e da fronteira da aplicação a ser medida. O escopo da contagem determina que funcionalidades serão consideradas para efeito do cálculo dos pontos. Normalmente, o cálculo é aplicado para todas as funções da aplicação ou do projeto. A fronteira da aplicação é uma questão chave, a fim de recuperar corretamente a propriedade dos dados que pertencem à aplicação, bem como para perceber o relacionamento da aplicação em estudo com sistemas externos.

A partir de então, o processo de cálculo propriamente dito é executado na três etapas a seguir: (i) *determinação dos pontos por função não-ajustados*; (ii) *cálculo do fator de ajuste*; e (iii) *ajuste do valor calculado*.

A Determinação dos Pontos por Função Não-ajustados (*Unadjusted Function Point – UFP*) reflete a funcionalidade entregue ao usuário, a partir dos módulos por ele requisitados e definidos. Os pontos por função não-ajustados contemplam as *funções de dados* e as *funções transacionais*. As funções de dados são:

- Arquivo Lógico Interno (*Internal Logical File* – ILF): são grupos de dados logicamente relacionados ou informações de controle, que sofrem manutenção pela própria aplicação.
- Arquivo de Interface Externa (*External Interface File* – EIF): são grupos de dados logicamente relacionados ou informações de controle, cujo processo de manutenção está sob a responsabilidade de outra aplicação.

As funções transacionais estão agrupadas da seguinte forma:

- Entrada Externa (*External Input* – EI): são processos elementares que tratam dados ou informações de controle, que entram pela fronteira da aplicação com o objetivo principal de efetuar manutenção nos ILF.
- Saída Externa (*External Output* – EO): são processos elementares que enviam informações de controle ou dados calculados para o usuário final ou outras aplicações.
- Consulta Externa (*External Inquiry* – EQ): são processos elementares que enviam informações de controle ou dados não calculados para o usuário final ou outras aplicações.

A segunda etapa corresponde ao Cálculo do Fator de Ajuste (*Value Adjustment Factor* – VAF), o qual é um assinalamento da funcionalidade geral provida ao usuário. O VAF é determinado a partir da soma dos fatores de influência de quatorze características gerais dos sistemas (*General System Characteristics* – GSCs).

A terceira e última etapa é o Ajuste do Valor Calculado na primeira etapa, cuja fórmula varia de acordo com o tipo de contagem. No caso de uma aplicação, por exemplo, multiplica-se o UFP pelo VAF.

As cinco seções a seguir irão discorrer sobre como se identificar e classificar as funções de dados ou transacionais específicas da aplicação sob avaliação, quando da execução da primeira etapa. Todas as definições, regras de contagem e classificação, tratamento de exceções e exemplos práticos que ilustram este processo podem ser encontrados em IFPUG (1999).

### 3.2. Identificação e Classificação dos Arquivos Lógicos Internos

Um Arquivo Lógico Interno (ILF) é formado por um grupo de dados logicamente relacionados ou informações de controle identificadas pelo o usuário, que sofrem manutenção dentro da fronteira da aplicação. Portanto, um ILF representa os requisitos de armazenamento de dados cujo processo de manutenção é executado pela própria aplicação.

Para identificar um ILF, é necessário constatar a veracidade dos seguintes preceitos:

- *Os dados são armazenados dentro da fronteira da aplicação;*
- *A manutenção desses dados é efetuada por um processo padrão da aplicação; e*
- *Os dados são identificados pelo usuário como sendo um requisito da aplicação.*

É fundamental que os dados sejam agrupados com base na visão do usuário, buscando-se sempre considerar os arquivos lógicos. Após a identificação, o ILF deve ser classificado de acordo com sua complexidade funcional relativa, que é baseada no número de Tipos de Elementos de Dados (DETs) e no número de Tipos de Elementos de Registro (RETs). Os DETs também são chamados de itens de dados e os RETs de registros lógicos.

Um RET pode ser definido como um subgrupo de dados relacionados logicamente dentro de um ILF ou EIF. Um subgrupo pode ser categorizado como opcional ou obrigatório em relação ao seu uso durante o processo, que cria as instâncias dos dados. Um DET é um campo não repetitivo, único, reconhecido pelo usuário e que possua significado próprio. Representa, desta forma, uma subdivisão do ILF ou EIF.

Um ILF deve ter sua complexidade classificada em *baixa*, *média* ou *alta*, de acordo com o número de DETs referenciados e com a quantidade de RETs identificados, segundo a *Tabela 3.1*.

Cada ILF classificado deve ser transformado em pontos por função não ajustados de acordo com a *Tabela 3.2*.

Tabela 3.1: Matriz de Complexidade de um ILF

Número de Registros Lógicos - RETs	Itens de Dados Referenciados – DETs		
	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

Tabela 3.2: Tabela de Tradução de um ILF

Faixa de Complexidade Funcional	Pontos por Função Não Ajustados
BAIXA	7
MÉDIA	10
ALTA	15

### 3.3. Identificação e Classificação dos Arquivos de Interface Externa

Um Arquivo de Interface Externa (EIF) é formado por um grupo de dados logicamente relacionados ou informações de controle identificadas pelo o usuário, que sofrem manutenção dentro da fronteira de outra aplicação. Portanto, um EIF mantém dados que são referenciados pela aplicação em medição, mas os processos que executam manutenção sobre os mesmos pertencem a outra aplicação. Percebe-se, assim, que um EIF identificado para uma aplicação deve ser um ILF em outra.

Para identificar um EIF, é necessário constatar a veracidade dos seguintes pontos:

- *Os dados são armazenados fora da fronteira da aplicação;*
- *Os dados não devem sofrer manutenção dentro da aplicação; e*
- *Os dados são identificados pelo usuário como sendo um requisito da aplicação.*

A diferença principal entre um ILF e um EIF é que este não é atingido pelos processos de manutenção dentro da fronteira da aplicação. Após a identificação, o EIF deve ser

classificado de acordo com sua complexidade funcional relativa, a qual é baseada no número de tipos de elementos de dados (DETs) e no número de tipos de elementos de registro (RETs), de forma semelhante ao processo aplicado para os ILFs.

Quando duas aplicações referenciam um mesmo ILF ou EIF, mas acessam conjuntos distintos de DETs, cada aplicação deve contabilizar apenas os DETs que ela própria referencia.

Um EIF deve ter sua complexidade classificado em *baixa*, *média* ou *alta*, de acordo com o número de DETs referenciados e com a quantidade de RETs identificados, segundo a *Tabela 3.3*.

Tabela 3.3: Matriz de Complexidade de um EIF

Número de Registros Lógicos - RETs	Itens de Dados Referenciados – DETs		
	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

Cada EIF classificado deve ser transformado em pontos por função não ajustados de acordo com *Tabela 3.4*.

Tabela 3.4: Tabela de Tradução de um EIF

Faixa de Complexidade Funcional	Pontos por Função Não Ajustados
BAIXA	5
MÉDIA	7
ALTA	10

### 3.4. Identificação e Classificação das Entradas Externas

Uma Entrada Externa (EI) desempenha as atividades de manutenção dos dados e de processamento do controle da aplicação. As informações de controle e dados processados entram pela fronteira da aplicação, através de um processo lógico único, com o objetivo

principal de efetuar manutenção sobre os ILFs ou de alterar o comportamento do sistema. Uma EI é uma transação executada diretamente pelo usuário ou por outra aplicação.

O primeiro procedimento na identificação das funções transacionais (EIs, EOs, EQs) é a identificação de processos elementares, observando as atividades que ocorrem dentro da aplicação. Um processo deve ser considerado como elementar quando obrigatoriamente satisfaz às duas condições abaixo:

1. *O processo é a menor unidade de atividade que possui significado para o usuário; e*
2. *O processo é uma transação que, ao finalizar, constata-se que o negócio da aplicação está consistente.*

Para que um processo elementar seja contabilizado como EI, torna-se necessário sua conformidade com as duas regras abaixo:

1. *O dado ou informação de controle entra pela fronteira da aplicação; e*
2. *Pelo menos um ILF sofre manutenção, caso o dado não seja uma informação de controle que altera o comportamento do sistema.*

Além das regras acima, pelo menos uma das três regras abaixo deve ser validada:

1. *A lógica do processo é diferente da lógica executada por outras EIs;*
2. *O conjunto dos dados é diferente dos outros conjuntos identificados para outras EIs; e*
3. *Os ILFs ou EIFs referenciados são diferentes dos arquivos referenciados por outras EIs.*

Após sua identificação, a EI deve ser classificada de acordo com sua complexidade funcional relativa, a qual baseia-se no número de arquivos referenciados (FTRs) e no número de itens de dados (DETs). O número de arquivos referenciados é a soma da quantidade de arquivos lógicos internos (ILFs) e da quantidade de arquivos de interface externa (EIFs) atualizados (somente ILFs) ou consultados na EI.

Para contabilizar os DETs, deve-se considerar as seguintes regras:



- Contar um DET para cada campo que entra ou sai pela fronteira da aplicação. Este campo deve ser não repetitivo, reconhecido pelo usuário e necessário para o processamento da EI;
- Não contar campos recuperados ou derivados pelo sistema, mesmo que sejam armazenados em algum ILF, caso não cruzem a fronteira da aplicação;
- Adicionar um DET para a capacidade de enviar mensagens de erro, confirmação de final de processamento ou verificação de continuidade de processamento; e
- Adicionar um DET para a habilidade de especificar uma ação a ser executada, seja através de botões, teclas de função ou outras formas, independente da quantidade de maneiras.

De acordo com a quantidade de arquivos e itens de dados referenciados, atribui-se uma complexidade *baixa*, *média* ou *alta* à EI, de acordo com a *Tabela 3.5*.

Tabela 3.5: Matriz de Complexidade de uma EI

Número de Arquivos Referenciados – FTRs	Itens de Dados Referenciados – DETs		
	1 a 4	5 a 15	16 ou mais
1	BAIXA	BAIXA	MÉDIA
2	BAIXA	MÉDIA	ALTA
3 ou mais	MÉDIA	ALTA	ALTA

Cada EI classificada deve ser vertida em pontos por função não ajustados, segundo a *Tabela 3.6*.

Tabela 3.6: Tabela de Tradução de uma EI

Faixa de Complexidade Funcional	Pontos por Função Não Ajustados
BAIXA	3
MÉDIA	4
ALTA	6

### 3.5. Identificação e Classificação das Saídas Externas

Uma Saída Externa (EO) é um processo elementar que tem como resultado o envio de dados ou informações de controle para fora da fronteira da aplicação. O processamento lógico de uma EO deve conter pelo menos uma fórmula matemática ou criar dados derivados.

Considerando-se que o processo elementar já foi identificado, conforme a *Seção 3.4*, deve-se aplicar as regras abaixo, para deduzir que o mesmo se trata de uma nova saída externa:

1. *A função envia dados ou informações de controle para fora da fronteira da aplicação; e*
2. *Pelo menos uma das seguintes condições é satisfeita:*
  - O processamento lógico é diferente do processamento das outras EOs;
  - O conjunto de dados é diferente dos outros conjuntos identificados para outras EOs; e
  - Os ILFs ou EIFs referenciados são diferentes dos arquivos referenciados por outras EOs.

Além das regras acima, pelo menos uma das três abaixo deve ser validada:

- *O processamento lógico contém pelo menos uma fórmula matemática;*
- *O processamento lógico cria dados derivados;*
- *O processamento lógico efetua manutenção sobre pelo menos um ILF; e*
- *O processamento lógico altera o comportamento da aplicação.*

Após sua identificação, a EO deve ser classificada de acordo com sua complexidade funcional relativa, a qual se baseia no número de arquivos referenciados (FTRs) e no número de itens de dados (DETs).

Para contabilizar os DETs, deve-se considerar as seguintes regras:

- Contar um DET para cada campo não repetitivo, reconhecido pelo usuário, que entra pela fronteira da aplicação e é necessário para especificar quando ou como o dado deve ser recuperado ou gerado pelo processo;
- Contar um DET para cada campo não repetitivo, reconhecido pelo usuário que sai pela fronteira da aplicação;
- Se um DET entra e sai pela fronteira da aplicação, este deve ser contado apenas uma vez;
- Não contar campos recuperados ou derivados pelo sistema, mesmo que sejam armazenados em algum ILF, caso não cruzem a fronteira da aplicação;
- Adicionar um DET para a capacidade de enviar mensagens de erro, confirmação de final de processamento ou verificação de continuidade de processamento;
- Adicionar um DET para a habilidade de especificar uma ação a ser executada, seja através de botões, teclas de função ou outras formas, independente da quantidade de maneiras;
- Não contar literais como títulos, cabeçalhos etc.; e
- Não contar variáveis como número de páginas, data, hora etc.

De acordo com a quantidade de arquivos e itens de dados referenciados, atribui-se uma complexidade *baixa*, *média* ou *alta* à EO, como pode ser visto na *Tabela 3.7*.

Tabela 3.7: Matriz de Complexidade de uma EO

Número de Arquivos Referenciados – FTRs	Itens de Dados Referenciados – DETs		
	1 a 5	6 a 19	20 ou mais
0 ou 1	BAIXA	BAIXA	MÉDIA
2 a 3	BAIXA	MÉDIA	ALTA
4 ou mais	MÉDIA	ALTA	ALTA

Cada EO classificada deve ser transformada em pontos por função não ajustados, segundo a *Tabela 3.8*.

Tabela 3.8: Tabela de Tradução de uma EO

Faixa de Complexidade Funcional	Pontos por Função Não Ajustados
BAIXA	4
MÉDIA	5
ALTA	7

### 3.6. Identificação e Classificação das Consultas Externas

Uma Consulta Externa (EQ) é um processo elementar, que envia dados ou informações de controle para fora da fronteira da aplicação. O processamento lógico não contém cálculos matemáticos nem cria dados derivados. Uma EQ não atualiza, portanto, um ILFs.

Considerando-se que o processo elementar já foi identificado conforme a *Seção 3.4*, deve-se aplicar todas as regras abaixo, para deduzir que o mesmo se trata de uma nova consulta externa:

1. *A função envia dados ou informações de controle para fora da fronteira da aplicação; e*
2. *Pelo menos uma das seguintes condições deve ser satisfeita:*
  - O processamento lógico é diferente do processamento das outras EQs;
  - O conjunto de dados é diferente dos outros conjuntos identificados para outras EQs; e
  - Os ILFs ou EIFs referenciados são diferentes dos arquivos referenciados por outras EQs.

Além das regras acima, pelo menos uma das três abaixo deve ser validada:

- *O processamento lógico recupera dados ou informações de controle de um ILF ou EIF;*
- *O processamento lógico não contém fórmulas matemáticas;*
- *O processamento lógico não cria dados derivados;*

- *O processamento lógico não efetua manutenção sobre ILFs; e*
- *O processamento lógico não altera o comportamento da aplicação.*

Após sua identificação, a EQ deve ser classificada de acordo com sua complexidade funcional relativa, que se baseia no número de arquivos referenciados (FTRs) e no número de itens de dados (DETs).

Para contabilizar os DETs, deve-se considerar as seguintes regras:

- *Contar um DET para cada campo não repetitivo, reconhecido pelo usuário, que entra pela fronteira da aplicação e é necessário para especificar quando ou como o dado deve ser recuperado ou gerado pelo processo;*
- *Contar um DET para cada campo não repetitivo, reconhecido pelo usuário que sai pela fronteira da aplicação;*
- *Se um DET entra e sai pela fronteira da aplicação, este deve ser contado apenas uma vez;*
- *Não contar campos recuperados ou derivados pelo sistema, mesmo que esses campos sejam armazenados em algum ILF, caso não cruzem a fronteira da aplicação;*
- *Adicionar um DET para a capacidade de enviar mensagens de erro, a confirmação de final de processamento ou a verificação de continuidade de processamento;*
- *Adicionar um DET para a habilidade de especificar uma ação a ser executada, seja através de botões, teclas de função ou outras maneiras, independente da quantidade das formas existentes;*
- *Não contar literais como títulos, cabeçalhos, etc.; e*
- *Não contar variáveis como número de páginas, data, hora, etc.*

De acordo com a quantidade de arquivos e itens de dados referenciados, atribui-se uma complexidade *baixa*, *média* ou *alta* à EQ, segundo a *Tabela 3.9* abaixo.

Tabela 3.9: Matriz de Complexidade de uma EQ

Número de arquivos referenciados – FTRs	Itens de Dados Referenciados – DETs		
	1 a 5	6 a 19	20 ou mais
0 ou 1	BAIXA	BAIXA	MÉDIA
2 a 3	BAIXA	MÉDIA	ALTA
4 ou mais	MÉDIA	ALTA	ALTA

Cada EQ classificada deve ser convertida em pontos por função não ajustados, segundo a *Tabela 3.10*.

Tabela 3.10: Tabela de Tradução de uma EQ

Faixa de Complexidade Funcional	Pontos por Função Não Ajustados
BAIXA	3
MÉDIA	4
ALTA	6

O total de pontos por função não ajustados será calculado pelo somatório de todos os pontos não ajustados das funções de dados e transacionais. A próxima seção apresenta o cálculo do fator de ajuste.

### 3.7. Cálculo do Fator de Ajuste

A segunda etapa na contagem dos pontos por função de uma aplicação trata do Cálculo do Fator de Ajuste (*Value Adjustment Factor* – VAF). O VAF baseia-se em quatorze características gerais do sistema (*General System Characteristics* – GSCs), buscando quantificar a funcionalidade geral da aplicação.

Cada uma dessas características deve ser avaliada de acordo com seu nível de influência (*Degree of Influence* – DI), que varia de 0 a 5, como descrito abaixo:

0 – *Sem influência*;

1 – *Influência incidental*;

- 2 – *Influência Moderada*;
- 3 – *Influência Média*;
- 4 – *Influência Significativa*; e
- 5 – *Forte influência*.

A identificação do nível de influência de cada uma das características sobre a aplicação é direcionada por um texto, que descreve cada grau de influência. Se nenhum dos textos refletir exatamente a situação em análise, deve-se escolher o grau que mais se aproxime do caso em questão. Os valores individuais das características devem ser guardados para comparações históricas e para facilitar o entendimento de diferentes projetos (Lokan, 1998).

Em seguida, os níveis de influência individuais devem ser somados para a obtenção do Nível de Influência Geral – TDI (*Total degree of influence*). O VAF é calculado pela seguinte equação:

$$\text{VAF} = (\text{TDI} * 0,01) + 0,65$$

O VAF tem o objetivo de avaliar o ambiente e a complexidade do processamento de sistema como um todo (Abran, 1996), constituindo-se numa tentativa de melhorar a relação pontos por função e esforço de projeto (Lokan, 1998).

Seguem-se essas quatorze características: *comunicação de dados, processamento distribuído, desempenho, utilização do equipamento, volume de transações, entrada de dados on-line, eficiência do usuário final, atualização on-line, processamento complexo, reutilização, facilidade de instalação, facilidade operacional, multiplicidade de locais, e facilidade de mudanças*.

### **3.7.1. Comunicação de Dados**

A *comunicação de dados* apresenta o grau de comunicação entre a aplicação e o processador, segundo a *Tabela 3.11*. Os dados e as informações de controle usados pela aplicação são enviados ou recebidos através de meios de comunicação. Terminais conectados localmente à unidade de controle utilizam recursos de comunicação. Todos os *links* de comunicação de dados requerem algum tipo de protocolo, que é um conjunto de regras ou convenções que permitem a troca ou intercâmbio de informações entre dois sistemas ou dispositivos.

Tabela 3.11: Comunicação de Dados

Grau de Influência	Descrição para Determinar o Grau de Influência
0	A aplicação é puramente <i>batch</i> ou executada em um PC <i>stand alone</i> .
1	A aplicação é <i>batch</i> , mas possui entrada de dados remota ou impressão remota.
2	A aplicação é <i>batch</i> , mas possui entrada de dados remota e impressão remota.
3	Há captura de dados <i>on-line</i> através de terminal de vídeo ou um <i>front-end</i> com o objetivo de alimentar processos <i>batch</i> ou sistemas de consultas.
4	A aplicação é mais que um <i>front-end</i> , porém suportando apenas um tipo de protocolo de comunicação.
5	A aplicação é mais que um <i>front-end</i> , suportando mais de um tipo de protocolo de comunicação.

### 3.7.2. Processamento Distribuído

O *processamento distribuído* mostra o grau com que a aplicação transfere dados entre os vários componentes da aplicação, segundo a *Tabela 3.12*.

Os dados ou funções de processamento distribuídos devem ser considerados dentro da fronteira da aplicação em análise.

Tabela 3.12: Processamento Distribuído

Grau de Influência	Descrição para Determinar o Grau de Influência
0	A aplicação não auxilia na transferência de dados ou processamento entre os componentes do sistema.
1	A aplicação prepara dados para o usuário processar em outro componente do sistema, como planilhas eletrônicas, gerenciadores de banco de dados para PC.
2	Os dados são preparados e depois transferidos para outro componente para lá serem processados. O processamento não é executado pelo usuário final.
3	Há processamento distribuído e transferência de dados <i>on-line</i> apenas numa direção.
4	Há processamento distribuído e transferência de dados <i>on-line</i> , em ambas as direções.
5	As funções de processamento são dinamicamente processadas no componente mais apropriado do sistema.



### 3.7.3. Desempenho

O *desempenho* descreve o grau em que as considerações de tempo de resposta e a capacidade de geração de informação (*throughput*) influenciaram ou podem influenciar o desenvolvimento da aplicação, segundo a *Tabela 3.13*.

Os objetivos de desempenho da aplicação estabelecidos ou aprovados pelo usuário em relação ao tempo de resposta e à capacidade de geração de informação influenciam o projeto, o desenvolvimento, a instalação e o suporte da aplicação.

Tabela 3.13: Desempenho

<b>Grau de Influência</b>	<b>Descrição para Determinar o Grau de Influência</b>
0	Nenhuma exigência especial de desempenho foi fixada pelo usuário.
1	Requisitos de desempenho e do projeto foram estabelecidos e revisados, mas não foram necessárias ações especiais.
2	O tempo de resposta ou a capacidade de geração de informação ( <i>throughput</i> ) são críticos durante as horas de pico. Nenhum procedimento especial para utilização da CPU foi requerido. O prazo final do processamento é sempre o próximo dia útil.
3	O tempo de resposta ou <i>throughput</i> é crítico durante o horário de utilização da aplicação. Não foi necessário nenhum procedimento especial para a utilização da CPU. Os requisitos de prazo de processamento com os sistemas de interface são restritos.
4	Além do item anterior, os requisitos de desempenho estabelecidos pelo usuário são rigorosos o suficiente, para exigir tarefas de análise de desempenho na fase de projeto.
5	Além do item 4, foram usadas ferramentas de análise de desempenho nas fases de projeto, desenvolvimento ou implementação para alcançar os requisitos de desempenho estabelecidos pela aplicação.

#### 3.7.4. Utilização do Equipamento

A *utilização do equipamento* apresenta o grau em que as restrições de recursos computacionais influenciaram ou podem influenciar o desenvolvimento da aplicação, como é exposto na *Tabela 3.14*.

Uma configuração operacional que requer considerações especiais de projeto é uma característica da aplicação.

Tabela 3.14: Utilização do Equipamento

<b>Grau de Influência</b>	<b>Descrição para Determinar o Grau de Influência</b>
0	Não há restrições operacionais explícitas ou implícitas.
1	Há restrições operacionais, mas são menos restritivas que as aplicações típicas. Nenhum esforço extra é necessário para suplantá-las.
2	Algumas considerações especiais sobre tempo e segurança são necessárias.
3	Há requisitos específicos do processador para uma parte específica da aplicação.
4	Restrições operacionais específicas requerem restrições especiais na aplicação em nível de processador central ou dedicado.
5	Além do item anterior, há restrições especiais da aplicação nos componentes distribuídos do sistema.

#### 3.7.5. Volume de Transações

Esta característica descreve o nível em que o *volume de transações* influenciou o projeto, envolvendo o desenvolvimento, a instalação e o suporte da aplicação, segundo a *Tabela 3.15*.

Tabela 3.15: Volume de Transações

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Não há previsão de pico de transações.
1	Há previsão de pico de transações durante um período (mensal, quadrimestral, semestral etc).
2	Há previsão de picos semanais.
3	Há previsão de picos diários.
4	Altos volumes de transações foram estabelecidos pelo usuário ou acordos dos níveis de serviço são altos o suficiente, para a execução de tarefas de análise de desempenho na fase de projeto da aplicação.
5	Altos volumes de transações foram estabelecidos pelo usuário ou acordos dos níveis de serviço são altos o suficiente para a execução de tarefas de análise de desempenho fase de projeto e, adicionalmente, para a requisição de uso de ferramentas da análise de desempenho nas fases de projeto, desenvolvimento ou implantação.

### 3.7.6. Entrada de Dados On-Line

A *entrada de dados on-line* exibe o grau em que os dados são inseridos na aplicação através de transações interativas, segundo a *Tabela 3.16*. Entrada de dados on-line e funções de controle são fornecidas pela aplicação.

Tabela 3.16: Entrada de dados On-Line

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Todas as transações são processadas em modo <i>batch</i> .
1	1% a 7% das transações são entradas interativas de dados.
2	8% a 15% das transações são entradas interativas de dados.
3	16% a 23% das transações são entradas interativas de dados.
4	24% a 30% das transações são entradas interativas de dados.
5	Mais de 30% das transações são entradas interativas de dados.

### 3.7.7. Eficiência do Usuário Final

A *eficiência do usuário final* descreve o grau de consideração para fatores humanos e facilidade de uso para o usuário da aplicação medida, segundo a *Tabela 3.17*.

As funções on-line fornecidas enfatizam um projeto da aplicação voltado para a eficiência do usuário final. O projeto poderá incluir:

- *Facilidades de navegação (teclas de função, atalhos, menus gerados dinamicamente etc.);*
- *Menus;*
- *Ajuda on-line e documentação;*
- *Movimento automático do cursor;*
- *Barras de rolagem;*
- *Impressão remota via transações on-line;*
- *Teclas de função pré-definidas;*
- *Aplicações batch submetidas via transações on-line;*
- *Seleção de dados na tela via movimentação do cursor;*
- *Uso intenso de vídeo reverso, brilho intensificado, sublinhado, cores e outros recursos de vídeo;*
- *Impressão da documentação via hard copy em transações on-line;*
- *Interface para mouse;*
- *Janelas pop-up;*
- *Uso mínimo possível para executar as funções do negócio;*
- *Suporte bilíngue (contar como quatro itens); e*
- *Suporte multilíngue (neste caso, considerar o suporte a mais de duas linguagens - contar como 6 itens).*

Tabela 3.17: Eficiência do Usuário Final

<b>Grau de Influência</b>	<b>Descrição para Determinar o Grau de Influência</b>
0	A aplicação não apresenta nenhum dos itens acima.
1	A aplicação apresenta de 1 a 3 dos itens acima.
2	A aplicação apresenta de 4 a 5 dos itens acima.
3	A aplicação apresenta 6 ou mais dos itens acima, mas não há nenhum requisito do usuário relacionado à eficiência.
4	A aplicação apresenta 6 ou mais dos itens acima, e os requisitos estabelecidos para a eficiência do usuário são rigorosos o suficiente, para que a fase de projeto da aplicação considere a inclusão de fatores humanos.
5	A aplicação apresenta 6 ou mais dos itens acima, e os requisitos estabelecidos para a eficiência do usuário são rigorosos o suficiente para que seja necessário o uso de ferramentas e processos especiais para demonstrar que os objetivos de eficiência foram alcançados.

### 3.7.8. Atualização On-Line

A *atualização on-line* mostra o grau em que os ILFs são atualizados por processos on-line da aplicação, segundo a *Tabela 3.18*.

Tabela 3.18: Atualização On-Line

<b>Grau de Influência</b>	<b>Descrição para Determinar o Grau de Influência</b>
0	Não há atualização on-line.
1	Há atualização on-line de 1 a 3 arquivos de controle. O volume de atualizações é baixo, e a recuperação de dados é simples.
2	Há atualização on-line de 4 ou mais arquivos de controle. O volume de atualizações é baixo, e a recuperação de dados é simples.
3	A maioria dos ILFs é atualizada de forma on-line.
4	Além dos itens anteriores, a proteção contra perda de dados é essencial e foi especificamente projetada e codificada no sistema.
5	Além dos itens anteriores, altos volumes de dados provocam considerações

	sobre custo no processo de recuperação. Há procedimentos automatizados de recuperação com intervenção mínima do operador.
--	---

### 3.7.9. Processamento Complexo

O *processamento complexo* descreve o grau em que o processamento lógico influenciou o desenvolvimento da aplicação, segundo a *Tabela 3.19*. Pode ser apresentado através dos cinco componentes abaixo:

- *Controles para fins especiais como auditoria e/ou processamento de segurança específico para a aplicação;*
- *Processamento lógico extensivo;*
- *Processamento matemático extensivo;*
- *Muito processamento de exceções, o que resulta em transações incompletas que necessitam de reproprocessamento, como, por exemplo, transações ATM incompletas causadas por interrupções de teleprocessamento, ausência de valores de dados ou falhas de validações etc.; e*
- *Processamento complexo para manipular múltiplas possibilidades de entrada/saída, como, por exemplo, multimídia ou independência de equipamentos.*

Tabela 3.19: Processamento Complexo

<b>Grau de Influência</b>	<b>Descrição para Determinar o Grau de Influência</b>
0	A aplicação não apresenta nenhum dos itens acima.
1	A aplicação apresenta um dos itens acima.
2	A aplicação apresenta 2 dos itens acima.
3	A aplicação apresenta 3 dos itens acima.
4	A aplicação apresenta 4 dos itens acima.
5	A aplicação apresenta os 5 itens acima.

### 3.7.10. Reutilização

A *reutilização* descreve o grau em que a aplicação e o código correspondente foram especificamente projetados, desenvolvidos e mantidos para serem reutilizados em outras aplicações, segundo a *Tabela 3.20*.

Tabela 3.20: Reutilização de Código

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Não apresenta código reutilizável.
1	O código reutilizável é usado somente dentro da própria aplicação.
2	Menos de 10% dos módulos considerou as necessidades de mais de um usuário.
3	10% ou mais da aplicação considerando-se sua utilização por outras aplicações.
4	A aplicação foi projetada e/ou empacotada para facilitar a reutilização, e a aplicação é <i>customizada</i> pelo usuário em nível de código fonte.
5	A aplicação foi projetada e/ou empacotada para facilitar a reutilização, e a aplicação é <i>customizada</i> para uso através de parâmetros que podem ser atualizados pelo usuário.

### 3.7.11. Facilidade de Instalação

A *facilidade de instalação* descreve o grau em que a conversão a partir de ambientes anteriores influenciou o desenvolvimento da aplicação, segundo a *Tabela 3.21*.

A facilidade de implantação e a conversão de dados são características da aplicação. Um plano de conversão e instalação e, possivelmente, ferramentas de conversão foram preparados e testados durante a fase de teste da aplicação.

Tabela 3.21: Facilidade de Implantação

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Não houve considerações especiais estabelecidas pelo usuário e nenhum procedimento especial é necessário para a implantação.
1	Não houve considerações especiais estabelecidas pelo usuário, mas um procedimento especial foi necessário para a implantação.
2	O usuário estabeleceu requisitos de implantação e conversão de dados, e roteiros de implantação foram preparados e testados. O impacto da conversão de dados no projeto não é considerado importante.
3	O usuário estabeleceu requisitos de implantação e conversão de dados, e roteiros de implantação foram preparados e testados. O impacto da conversão de dados no projeto é considerado importante.
4	Além do item 2 acima descrito, ferramentas automáticas de implantação e conversão foram fornecidas e testadas.
5	Além do item 3 acima descrito, ferramentas automáticas de implantação e conversão foram fornecidas e testadas.

### 3.7.12. Facilidade Operacional

A *facilidade operacional* mostra o grau que a aplicação atende a aspectos operacionais, como inicialização, *backup* e processos de recuperação, conforme a *Tabela 3.22*.

A facilidade operacional é uma característica da aplicação. A aplicação minimiza a necessidade de atividades manuais, tais como montagem de fitas magnéticas, manuseio de formulários e intervenção do operador.



Tabela 3.22: Facilidade Operacional

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Não houve considerações especiais estabelecidas pelo usuário a não ser os procedimentos normais de <i>backup</i> .
1 – 4	<p>Selecionar os itens abaixo que se aplicam ao sistema. Se nada for dito, cada item possui o valor de 1 ponto:</p> <ul style="list-style-type: none"> <li>• Procedimentos efetivos de inicialização, <i>backup</i> e recuperação foram preparados, mas é preciso a intervenção manual do operador;</li> <li>• Procedimentos efetivos de inicialização, <i>backup</i> e recuperação foram preparados, e nenhuma intervenção do operador é preciso (contar 2 itens);</li> <li>• A aplicação minimiza a necessidade de montagem de fitas; e</li> <li>• A aplicação minimiza a necessidade de manipulação de papéis.</li> </ul>
5	A aplicação foi projetada para não precisar de intervenção do operador, além da inicialização e finalização da aplicação. A recuperação automática de erros é uma característica da aplicação.

### 3.7.13. Multiplicidade de Locais

A aplicação foi especificamente projetada, desenvolvida e mantida para ser instalada em *múltiplos locais* ou em mais de uma organização, segundo a *Tabela 3.23*.

Tabela 3.23: Multiplicidade de Locais

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Não houve nenhuma solicitação do usuário para considerar a necessidade de instalar a aplicação em mais de um local.
1	Houve a necessidade de considerar no projeto do sistema a instalação da aplicação em múltiplos locais. A aplicação foi projetada para operar somente em ambientes idênticos de hardware e software.
2	Houve a necessidade de considerar no projeto do sistema a instalação da aplicação em múltiplos locais. A aplicação foi projetada para operar

	somente em ambientes similares de hardware e software.
3	Houve a necessidade de considerar no projeto do sistema a instalação da aplicação em múltiplos locais. A aplicação foi projetada para operar inclusive em ambientes diferentes de hardware e software.
4	Além de atender aos itens 1 e 2, um plano de documentação e manutenção foi elaborado e testado para manter a aplicação em múltiplos locais.
5	Além de atender ao item 3, um plano de documentação e manutenção foi elaborado e testado para manter a aplicação em múltiplos locais.

### 3.7.14. Facilidade de Mudanças

A *facilidade de mudanças* descreve o grau em que uma aplicação foi desenvolvida com o objetivo de facilitar a modificação do processamento lógico ou a estrutura de dados, segundo a *Tabela 3.24*.

As seguintes características devem ser analisadas para se descobrir que facilidades podem ser atribuídas à aplicação:

- *São fornecidos recursos que provêem facilidade de relatório e de consulta flexível para manipular pedidos simples, como, por exemplo, lógica and/or aplicada a somente um ILF (contar como 1 item);*
- *São fornecidos recursos que provêem facilidade de relatório e de consulta flexível para manipular pedidos de consulta de média complexidade, como, por exemplo, lógica and/or aplicada a mais de um ILF (contar como 2 itens);*
- *São fornecidos recursos que provêem facilidade de relatório e consulta flexível para manipular pedidos complexos; por exemplo, combinações de lógica and/or aplicadas a um ou mais ILFs (contar como 3 itens);*
- *Dados de controle do negócio são mantidos pelo usuário através de processos on-line interativos, mas as alterações só são efetivadas no próximo dia útil; e*
- *Dados de controle do negócio são mantidos pelo usuário através de processos on-line interativos, e as alterações têm efeito imediato (contar como 2 itens).*

Tabela 3.24: Facilidade de Mudanças

Grau de Influência	Descrição para Determinar o Grau de Influência
0	Nenhum item foi contabilizado.
1	O total de 1 item foi contabilizado.
2	O total de 2 itens foi contabilizado.
3	O total de 3 itens foi contabilizado.
4	O total de 4 itens foi contabilizado.
5	O total de 5 itens foi contabilizado.

A terceira etapa, cálculo dos pontos ajustados, está apresentada nas próximas seções, conforme o tipo de contagem.

### 3.8. Cálculo dos Pontos por Função Ajustados - Aplicação

Os pontos por função da aplicação – AFP (*Application Function Point*) – são calculados pela multiplicação dos pontos não ajustados (UFP) pelo VAF:

$$AFP = UFP * VAF$$

### 3.9. Cálculo dos Pontos por Função Ajustados – Projeto de Desenvolvimento

Os pontos por função de um projeto de desenvolvimento (*Development Project Function Point* – DFP) podem ser calculados de forma similar a uma aplicação, devendo-se considerar adicionalmente os pontos referentes ao processo de conversão (*Conversion Unadjusted Function Point* - CFP), ou seja, funções que serão criadas durante o desenvolvimento do sistema, mas que desaparecerão com a implantação da aplicação:

$$DFP = (UFP + CFP) \times VAF$$

### 3.10. Cálculo dos Pontos por Função Ajustados – Projeto de Manutenção

Os pontos por função de um projeto de manutenção (*Enhancement Project Function Point* – EFP) são obtidos pela equação abaixo:

$$EFP = [(ADD + CHGA + CFP) \times VAFA] + (DEL \times VAFB)$$

onde,

- *ADD* são os pontos por função não ajustados das funções adicionadas na aplicação pelo projeto de manutenção;
- *CHGA* são os pontos por função não ajustados das funções modificadas pelo projeto de manutenção, depois que as alterações forem aplicadas;
- *VAFA* é o fator de ajuste da aplicação após a manutenção;
- *DEL* corresponde aos pontos por função não ajustados das funções excluídas da aplicação; e
- *VAFB* é o fator de ajuste da aplicação antes da manutenção.

Após a manutenção ser concluída, a aplicação deve ser recalculada através da seguinte forma:

$$AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] \times VAFA$$

onde,

- *UFPB* são os pontos por função não ajustados da aplicação antes da manutenção; e
- *CHGB* são os pontos por função não ajustados das funções modificadas pelo projeto de manutenção, antes das modificações terem sido aplicadas.

A próxima seção discorre sobre os pontos fortes e fracos da FPA.

### 3.11. Características Primordiais da FPA

A técnica de FPA vem sendo mundialmente utilizada nas organizações (Fowler, 1999), alcançando comprovado sucesso (Symons, 2001). Adicionalmente, vários institutos de

pesquisa desenvolvem estudos, citando como pontos positivos da técnica, que ultrapassam o objetivo da medição:

- *Auxilia os gerentes a entenderem o impacto das requisições de mudanças e melhorias* (Simon, 2000);
- *Mostra tendências de produtividade* (Simon, 2000);
- *Provê uma medida de tamanho para suportar qualidade* (Ho, 1999), *sendo possível, por exemplo, calcular a quantidade de erros por ponto por função*;
- *Auxilia no gerenciamento de mudança de escopo do projeto* (Heller, 1996);
- *Pode ser utilizada como ferramenta de comunicação com os usuários, fazendo com estes estejam mais envolvidos* (Postion, 1999) *e expondo requisitos técnicos e funcionais* (Heller, 1996);
- *Independente do estilo de codificação e da arquitetura do software* (Vandoren, 1997);
- *As funções de dados servem para validar o modelo de dados e detectar dados ausentes* (Postion, 1999); e
- *Pode ser usada para melhorar processos* (Miccolis, 2001).

No entanto, muitas limitações da técnica foram evidenciadas. Algumas se mostram como aspectos negativos que devem ser cuidadosamente analisados, antes de optar por medir o tamanho de um projeto através da FPA. As que mais se destacam são:

- *Mostra-se inadequada para aplicações com ênfase em funções e controles (sistemas de engenharia), visto que a complexidade de tais aplicações se concentra nos algoritmos* (Pressman, 2000);
- *Permite subjetividade durante sua aplicação, quando se está identificando as funções* (Simon, 2000);
- *Não mede precisamente sistemas de tempo real* (Dumke, 1999), *sistemas operacionais, software embutido e software de comunicação* (Jones, 1995);
- *Depois que o produto está desenvolvido, torna-se difícil levantar as informações necessárias para uma contagem precisa* (Harrison, 2001);

- *Não atende aos critérios de (Zuse, 1992; Kitchenham, 1995) sobre medida de software válida, pois soma medidas de escala ordinal e é descontínua nos limites inferiores (Rudolph, 1997);*
- *Não está em conformidade com a norma ISO 14143-1, pois a mesma requer que não haja passos de ajuste (Dekkers, 1998). No caso da FPA, os pontos por função são ajustados através do VAF;*
- *Viola o Axioma da Dominância, pois nem sempre um acréscimo de funcionalidade aumenta o “tamanho” do projeto, e o Axioma da Monotonicidade, pois não é possível prever o tamanho funcional de uma aplicação inteira a partir dos valores obtidos através das medições de seus componentes (Fetcke, 1999);*
- *Há falta de ortogonalidade entre os elementos. Isto sugere que os mesmos elementos básicos podem estar sendo contados mais de uma vez e propõe a existência de um modelo de métrica mais simples (Lokan, 1999);*
- *Há dupla contagem de complexidade (Lokan, 1998), pois classifica a complexidade das funções e, posteriormente, ajusta tal complexidade através das características gerais do sistema;*
- *Algumas características, como entrada de dados on-line, estão desatualizadas (Lokan, 1998);*
- *O VAF é irrelevante para sistemas modernos (Symons, 2001);*
- *A tradução de termos como objetos, Use Cases e API's para conceitos como entradas externas e arquivos lógicos internos é difícil (Symons, 2001);*
- *Na realidade, requisitos de qualidade podem causar impacto maior do que 35%, que é o limite imposto pelas características gerais do sistema (Kitchenham, 2001); e*
- *As aplicações modernas atingem facilmente os níveis mais altos de complexidade durante a classificação das funções, não refletindo, portanto, verdadeiramente sua complexidade e, conseqüentemente, seu tamanho. Esta situação pode ser encontrada, por exemplo, em datawarehouse (Santillo, 2001) e sistemas governamentais (Lima, 2001a).*

A próxima seção descreve algumas das principais extensões da FPA, as quais buscam solucionar algumas das limitações acima citadas.

### **3.12 Extensões da Análise de Pontos por Função**

Para contornar algumas limitações da análise de pontos por função, foram propostas algumas extensões da mesma, principalmente para estimar sistemas com características proeminentes de complexidade, como os sistemas de tempo real ou equipamentos com software embutido.

Em 1986, a *Software Productivity Research, Inc.* (SPR) desenvolveu uma extensão da FPA chamada *Feature Points*, que foi publicada em Jones (1991). Esta métrica mostrou-se adequada para software que possuem algoritmos de alta complexidade, como sistemas de tempo real, equipamentos com software embutido, sistemas operacionais, sistemas de comunicação, sistemas de controle de processos, aplicações CAD, sistemas matemáticos etc.

A *Feature Points* introduz um novo parâmetro para estimar tamanho de software (algoritmo). Um algoritmo é definido como um conjunto de regras, que devem ser completamente expressas, para solucionar um problema computacional específico. Exemplos de algoritmo são: rotina para cálculo de raiz quadrada, inversão de matrizes e outras.

Conceitualmente, *feature points* e pontos por função são bastante semelhantes. Isto é comprovado quando se aplicam as técnicas a sistemas clássicos ou gerenciais, pois ambas retornam valores aproximados. Porém, quando o sistema avaliado possui algoritmos complexos, a *feature points* normalmente produz valores mais elevados.

De forma semelhante, a *Full Function Point* (FFP, 1997) é uma recente tentativa de estender a FPA, introduzida pela Universidade de Quebec, para fornecer maior fidelidade aos sistemas de tempo real e equipamentos com software embutido. A maioria das regras de contagem da FPA estão inclusas na FFP. Porém, a FFP acrescenta duas novas funções de dados de controle (UCG – Grupo de Controle de Atualização e RCG – Grupo de Controle de Leitura) e quatro novas funções transacionais de controle (ECE – Entrada de Dados Externos de controle, ECX- Saída de Dados Externos de Controle, ICR – Leitura de Dados Internos de Controle e ICW – Gravação de Dados Internos de Controle).

A versão 2.0 da FFP irá se tornar a COSMIC 1.0 (Symons, 1999). Outras extensões como a *3D Function Points* e a *Mark II Function Points* podem ser encontradas em (Whitmire, 1995) e (Symons, 1988), respectivamente.

Já a métrica *Use Case Points* é uma modificação da FPA, resultante das pesquisas de Karner (1993). Considerando que a análise do sistema foi elaborada através de casos de uso, o prazo para o desenvolvimento do projeto pode ser estimado nos estágios iniciais do projeto. Em linhas gerais, o cálculo dos pontos segue os passos abaixo (Fihlman, 2000):

1. Inicialmente, os atores do projeto são classificados em *simples*, *médio* ou *complexo*, aferindo-lhes pesos a cada um deles;
2. De forma semelhante, os casos de uso são classificados em *simples*, *médio* ou *complexo*, de acordo com o número de transações;
3. Os valores obtidos nos passos 1 e 2 são somados, obtendo-se o total de pontos não-ajustados dos casos de uso (UUCP);
4. Em seguida, deve-se calcular a Complexidade Técnica do Sistema (TCF). A 13 fatores técnicos são atribuídos valores de 0 a 5. Da soma desses valores resulta o Fator Técnico Total (Tfactor). O TCF é obtido pela fórmula abaixo:

$$TCF = 0,6 + (0,01 * Tfactor)$$

5. De modo semelhante, calcula-se o Fator do Ambiente (EF), o qual considera as habilidades da equipe de desenvolvimento. Valores de 0 a 5 são atribuídos a 8 fatores ambientais, cuja soma fornece o fator ambiental total (Efactor). O EF é assim obtido:

$$EF = 1,4 + (-0,03 * Efactor)$$

6. O valor final dos *Use Case Points* é calculado da seguinte forma:

$$UCP = UUCP * TCF * EF$$

Originalmente, Karner (1993) propôs que cada UCP demandasse 20 homens-horas.



### 3.13 Conclusão

Este capítulo abordou a Análise de Pontos por Função (FPA), apresentando seus objetivos, características e principais conceitos de forma detalhada. Foram também mencionadas as principais extensões já propostas para esta técnica.

A FPA é o método para medição de tamanho funcional bastante conhecido na literatura e muito utilizado nas organizações. Seus conceitos apresentam um fácil entendimento, o que beneficia o processo de aprendizagem. Além disso, sistemas convencionais, como por exemplo, sistemas de informações gerenciais, podem ter sua complexidade funcional avaliada em um curto espaço de tempo.

No entanto, está cada vez mais difícil manter a consistência na interpretação de novos termos, como *caso de uso* e, além disso, o incremento de novas regras pode tornar o processo muito complexo. Adicionalmente, não se imaginou na década de 70 que os sistemas atuais alcançariam porte tão expressivo. Este aspecto faz com que as estimativas de custo e prazo para desenvolvimento dos sistemas ultrapassem as margens de erro aceitas pelo mercado. Este cenário é propício para a criação de extensões da FPA e para o surgimento de novos métodos.

O próximo capítulo aborda os principais conceitos da teoria *fuzzy*, para fundamentar este trabalho, que é a extensão da análise de pontos por função utilizando conceitos da teoria *fuzzy*.