PFI
Pontos de Função de Implementação

Equipe

Mauricio Aguiar

Valter Braghin

Nelson Camargo

Irapuã da Costa Jr.

Gabriel Dalla Lana

Elina Madeira

Versão 1.00
22/04/2004

Conteúdo

1 Introdução

Este trabalho tem como objetivo a definição e aplicação de uma medida de tamanho de software voltada para a implementação. Essa medida foi denominada “Pontos de Função de Implementação”, ou PFI.

Diferentemente dos Pontos de Função (PF) convencionais, que medem o software a partir da perspectiva do usuário final do negócio, os PFI pretendem capturar o tamanho da aplicação a partir do ponto de vista do desenvolvedor, ou seja, de uma perspectiva centrada na implementação.

Os PFI foram concebidos para oferecerem, ao mesmo tempo, uma alternativa para a medição funcional do software a partir da perspectiva da implementação e manterem máxima compatibilidade com os conceitos e terminologia do IFPUG.

A correspondência ou mapeamento dos conceitos estabelecidos neste trabalho com os elementos computacionais de cada ambiente específico deverá ser realizado pelos respectivos usuários.

2 Definições

2.1 Perspectivas do Usuário Final e da Implementação

Inicialmente, fazemos distinção entre a perspectiva do usuário final e a perspectiva da implementação.

A perspectiva do usuário final considera apenas a funcionalidade presente no software aplicativo que deve ser entregue para satisfazer os requisitos do negócio. Esta é a perspectiva utilizada nas medições realizadas com base nos Pontos de Função do IFPUG. Não é o mesmo que "visão do usuário"[1].

A perspectiva da implementação considera toda a funcionalidade presente em cada parte do software, que deve ser desenvolvida ou implementada para satisfazer os requisitos do negócio. Esta será a perspectiva utilizada nas medições realizadas com base nos PFI[2].

2.2 Usuário, Unidade de Software, Fronteira e Aplicação

Para medir a funcionalidade presente na implementação é fundamental definir a fronteira que irá separar cada unidade de software dos demais objetos do modelo. Para definir a fronteira, será necessário introduzir o conceito de usuário. Para identificar a fronteira, será necessário estabelecer os conceitos de estímulo, dado e informação de controle.

Uma unidade de software (US) é o menor "pedaço" de software para o qual deverá ser realizada medição. Uma US é delimitada por uma fronteira.

A fronteira indica o limite entre a unidade de software sendo medida e o usuário, definindo o que faz e o que não faz parte da US[3].

Uma aplicação é um todo coeso constituído por uma ou mais unidades de software.

Um usuário é qualquer pessoa que especifica requisitos funcionais para o software, e/ou qualquer pessoa ou objeto que se comunica ou interage com o software a qualquer tempo. Pode ser um ser humano, outro software, um dispositivo de hardware, etc, desde que especificado nos requisitos funcionais[4].

Um estímulo é uma representação de um evento que ocorre fora da fronteira de uma unidade de software e que provoca o início de um processo dentro da US. É o conceito de "evento" aplicado à implementação. Um evento "externo" pode dar origem a vários estímulos "internos" à aplicacão. Um estímulo é comunicado à unidade de software através da entrada de dados, informações de controle, ou do passar do tempo[5].

Um dado é o valor de um atributo de uma entidade ou objeto de interesse de uma unidade de software. O que é um dado para uma unidade de software depende do nível de abstração dessa US.

---

[1] "User View" – IFPUG. *Counting Practices Manual Version 4.1.1* - Princeton Junction, 2000.
[2] "Developer Viewpoint" – COSMIC. Measurement Manual Version 2.2 – January 2003.
[3] "Boundary" – ISO. *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts*. – ISO 14143:1998. Genève: ISO/IEC, 1998.
[4] "User" - ISO. *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts*. – ISO/IEC 14143:1998. Genève: ISO/IEC, 1998.
[5] "Stimulation" – Clements, P. et al. *Documenting Software Architectures* - Boston: Addison-Wesley, 2003. Pg. 264.

Uma informação de controle dispara ou influencia um processo de uma unidade de software. Informações de controle determinam o quê, como, ou quando os dados serão processados[6].

2.3 Visão do Usuário

Na contagem de PFI, a visão do usuário é um dos conceitos que difere do utilizado nas contagens de Pontos de Função do IFPUG. O usuário, no caso dos PFI, é um observador situado "dentro" da unidade de software sendo considerada. Uma forma de compreender o ponto de vista desse usuário é identificar-se com o analista de sistemas, arquiteto ou programador encarregado de construir a unidade de software, supondo que o mesmo sabe apenas o indispensável para poder compreendê-la e construí-la.

2.4 Camadas de Software

Um conceito útil na definição das fronteiras, sob o ponto de vista da implementação, é o de camada de software (ou simplesmente camada). A definição de uma camada de software tem a ver com o nível de abstração.

Uma camada de software reflete a divisão do software em unidades, onde cada uma representa uma "máquina virtual". As camadas particionam o software completamente, de modo que cada camada oferece um conjunto de serviços coeso[7].

Abstração é o processo de análise do domínio do problema (mundo real) enfatizando fatores relevantes do processo de maneira a alcançar um modelo, ou o resultado desse processo[8]. Dizemos que as unidades de software operam em níveis de abstração, modelos que são constituídos por conceitos e regras dotados de significado específico. Um nível de abstração é considerado mais baixo que outro quando o respectivo modelo da realidade é mais específico, ou menos abrangente que o do outro.

2.5 Coesão e Acoplamento

Os conceitos de coesão e acoplamento foram bastante explorados pela Análise Estruturada[9].

A coesão mede o grau de homogeneidade funcional de uma unidade de software. As funções de uma unidade de software coesa podem ser consideradas como pertencentes a um mesmo grupo funcional.

O acoplamento mede o grau de independência de uma unidade de software em relação às demais. Uma US não acoplada possui baixo grau de dependência em relação às demais, excetuando-se aquelas mais baixas na hierarquia de camadas, cujos serviços espera-se que a US utilize.

2.6 Processo Elementar

Um processo elementar é a menor unidade funcional que ainda possui significado no nível de abstração da US que o contém. Um processo elementar é completo em relação ao nível de abstração da unidade de software[10].

---

[6] "Control Information" - IFPUG. *Counting Practices Manual Version 4.1.1* - Princeton Junction, 2000.
[7] "Layers" - Clements, P. et al. *Documenting Software Architectures* - Boston: Addison-Wesley, 2003. Pg. 77.
[8] "Abstraction" - COSMIC. Measurement Manual Version 2.2 – January 2003.
[9] "Coupling"; "Cohesion" – Pressman, Roger. Software Engineering – A Practitioner's Approach. 4th Edition. New York: McGraw-Hill, 1997. Pg. 357.
[10] "Elementary Process" - IFPUG. *Counting Practices Manual Version 4.1.1* - Princeton Junction, 2000.

Exemplo: para uma unidade de software "X", com a função "Efetuar Transferência entre Contas Correntes", a atividade "Fazer o débito na conta do correntista A" não seria considerada um processo elementar (completo) para essa unidade de software. Nesse nível de abstração, o processo só estaria completo após ter sido efetuado o crédito na conta corrente do correntista "B".

Já para uma unidade de software "Y", voltada exclusivamente para a atualização do banco de dados, a atividade "Fazer o débito na conta do correntista A" seria interpretada (possivelmente) como "Atualizar o campo Saldo com o valor X", que poderia ser um processo elementar (completo) para essa unidade. Na visão dessa unidade, receber um valor e atualizar um campo do banco de dados com esse valor constituiria um processo completo. Evidentemente, neste caso a unidade "Y" estaria em uma camada hierarquicamente inferior à da unidade "X".

Para que um processo elementar (PE) seja considerado único (não duplicado) em uma US, pelo menos uma das seguintes condições deve ser verdadeira:

- A lógica de processamento do PE deve ser diferente da lógica de processamento dos demais processos elementares da US;

- Os arquivos referenciados pelo PE devem ser diferentes dos arquivos referenciados pelos demais processos elementares da US;

- Os itens de dado referenciados pelo PE devem ser diferentes dos itens de dado referenciados pelos demais processos elementares da US.

2.7 Acesso Direto e Indireto

Dizemos que um arquivo (ALI ou AIE) é acessado diretamente por uma unidade de software (US) se a US utiliza, no acesso, exclusivamente software residente em sua própria fronteira, excetuando-se os recursos de software básico utilizados – sistema gerenciador de banco de dados, sistema de teleprocessamento, rede, sistema operacional, etc. Neste caso, o acesso é denominado direto.

Por outro lado, dizemos que um arquivo (ALI ou AIE) é acessado indiretamente por uma unidade de software se a US utiliza, no acesso, software residente em outras US, excetuando-se os recursos de software básico utilizados. Neste caso, o acesso é denominado indireto.

Exemplo: Uma organização criou uma classe denominada Cliente (no sentido O-O) específica para lidar com o objeto Cliente. Todo o acesso à tabela de Clientes é efetuado através dessa classe. Neste caso, diremos que a US constituída pela classe Cliente acessará diretamente o arquivo Clientes. Na organização em questão, todas as demais US acessarão indiretamente o arquivo Clientes, uma vez que só o farão através de software residente na US Cliente.

3 Procedimento de Contagem

O procedimento de contagem utilizado na medição com base em PFI é o mesmo definido pelo IFPUG para a contagem de Pontos de Função, conforme ilustra a Figura 1.

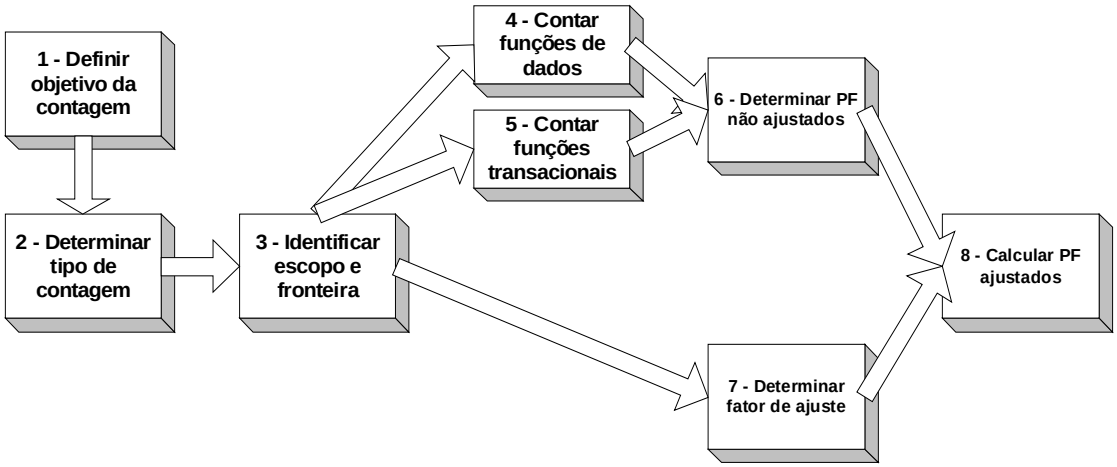

Figura 1 – Procedimento de Contagem de PFI

A seguir, detalhamos cada um dos passos do procedimento acima, segundo a visão dos PFI. O Exemplo Modelo ilustra cada um dos passos, sendo desenvolvido ao longo da explanação.

3.1 Definir o Objetivo da Contagem

O objetivo da contagem de PFI é resolver um problema do negócio[11]. A escolha dos PFI implica que a solução do problema exige que a aplicação seja considerada do ponto de vista da implementação, ou do desenvolvedor. Desta forma, a medida obtida como resultado da contagem não terá, necessariamente, significado claro para os usuários finais do negócio, mas sim para os desenvolvedores e outros envolvidos com a perspectiva da implementação.

Exemplo Modelo: O objetivo da contagem é: "Calcular a quantidade de PFI de um projeto de desenvolvimento, a fim de determinar o valor a ser pago aos respectivos fornecedores".

3.2 Determinar Tipo de Contagem

Não há diferença entre os tipos de contagem definidos para os PFI e os do IFPUG. Os tipos de contagem são três:

- Contagem de projeto de desenvolvimento
- Contagem de projeto de melhoria
- Contagem de aplicação

Exemplo Modelo: Trata-se de uma contagem de um projeto de desenvolvimento.

[11] "Purpose of the Count" - IFPUG. *Counting Practices Manual Version 4.1.1* - Princeton Junction, 2000.

3.3 Identificar Escopo e Fronteira

O escopo da contagem define a funcionalidade que será incluída na contagem. Define um sub-conjunto do software e depende do objetivo da contagem[12].

Exemplo Modelo: O escopo do projeto de desenvolvimento inclui todas as unidades de software criadas pelo projeto.

A fronteira indica o limite entre as unidades de software sendo medidas e seus usuários (ver definição acima). A determinação da fronteira no cálculo de PFI deverá obedecer aos seguintes critérios:

3.3.1 Determinar as Camadas de Software

A determinação das camadas de software auxilia na identificação das fronteiras. A seguir, fornecemos algumas dicas e orientações para o estabelecimento das camadas. Neste estágio, nenhuma das orientações apresentadas deve ser considerada como uma "regra absoluta".

3.3.1.1 Conceito de Máquina Virtual

Cada camada pode ser visualizada como sendo uma "máquina virtual", muitas vezes possuindo um processador específico a ela alocado[13].

3.3.1.2 Hierarquia

As camadas de software obedecem a um modelo hierárquico, no qual as camadas de nível mais alto utilizam os serviços daquelas de nível mais baixo. Se A e B são duas camadas podemos ter, por exemplo, a camada A em um nível superior a B. Neste caso, A poderá utilizar livremente todos os serviços públicos de B. Isso não será verdade em relação a B. Se B for uma camada inferior a A, B não poderá utilizar livremente os serviços de A.

Um exemplo simples envolve a camada de apresentação de uma aplicação, implementada em HTML – camada "A" e a camada residente no servidor web, implementada em ASP – camada "B". Neste caso, a camada "A" é hierarquicamente superior à camada "B": em outras palavras, "A" é cliente de "B".

3.3.1.3 Níveis de Abstração

Todas as unidades residentes na mesma camada operam no mesmo "nível de abstração". Por exemplo, é normal que a camada de apresentação (de nível mais alto) opere no nível de abstração do negócio – conheça a terminologia e os conceitos usados pelo negócio. O mesmo não poderia ser dito de um "device driver", cujo nível de abstração é diferente e "mais baixo".

Unidades de software residentes em camadas diferentes interpretarão os mesmos dados de forma diferente, cada qual de acordo com seu próprio modelo. Por exemplo, a camada de apresentação de uma aplicação web pode denominar um determinado grupo de dados "Dados do Cliente". Esses dados poderão ser passados a uma camada de "transporte" cujo objetivo seja apenas fazer a triagem e encaminhar os dados ao destino. Para esta camada, os dados

[12] "Scope of the Count" - IFPUG. *Counting Practices Manual Version 4.1.1* - Princeton Junction, 2000.

[13] "Layers" - Clements, P. et al. *Documenting Software Architectures* - Boston: Addison-Wesley, 2003. Pg. 77.

do cliente seriam conhecidos como “Pacote Recebido” (por exemplo), uma vez que o nível de abstração da mesma não reconheceria o conceito “cliente”.

O fato de duas unidades de software interpretarem os mesmos dados da mesma maneira é um indicador de que as mesmas residem em uma única camada. No entanto, este não é um critério conclusivo, como passamos a demonstrar. Uma unidade de software “A” construída em Visual Basic, residente na camada de aplicação, pode comunicar-se com uma *stored procedure* “B” programada em T-SQL, residente na camada de dados. Embora residam em camadas diferentes, “A” e “B” podem interpretar os dados trocados exatamente da mesma maneira. Por exemplo, “A” pode enviar um grupo de dados referentes a “Cliente” para “B” atualizar no banco de dados. “B” pode “enxergar” este grupo de dados da mesma forma que “A”.

Se duas unidades de software puderem usar livremente os serviços uma da outra, essas unidades serão normalmente consideradas como residentes na mesma camada e denominadas “pares” (peers).

3.3.1.4 Independência

Uma unidade residente em uma camada inferior pode operar independentemente das camadas superiores. Isto não é verdade em relação às camadas superiores.

3.3.1.5 Outras Considerações

Normalmente, consideraremos como estando em camadas diferentes: gerenciador de banco de dados, device drivers, sistema operacional, parte residente no cliente de uma aplicação web, camada residente em um servidor de dados atendendo a várias aplicações, camada de triagem e transporte de dados entre vários servidores, etc. Em aplicações comerciais a camada de nível mais alto corresponderá, normalmente, à US que faz interface com o usuário do negócio. O estilo de arquitetura utilizado pela organização deve servir de base à definição das camadas. Os conceitos de coesão e acoplamento, conforme definidos por Pressman[14], são ferramentas conceituais auxiliares na identificação das camadas. O claro estabelecimento das camadas é um importante pré-requisito para a utilização dos PFI.

Exemplo Modelo: As camadas de software envolvidas neste projeto são: Camada de Apresentação, Camada de Validação e Regras de Negócio e Camada de Dados. Ver Figura 2.

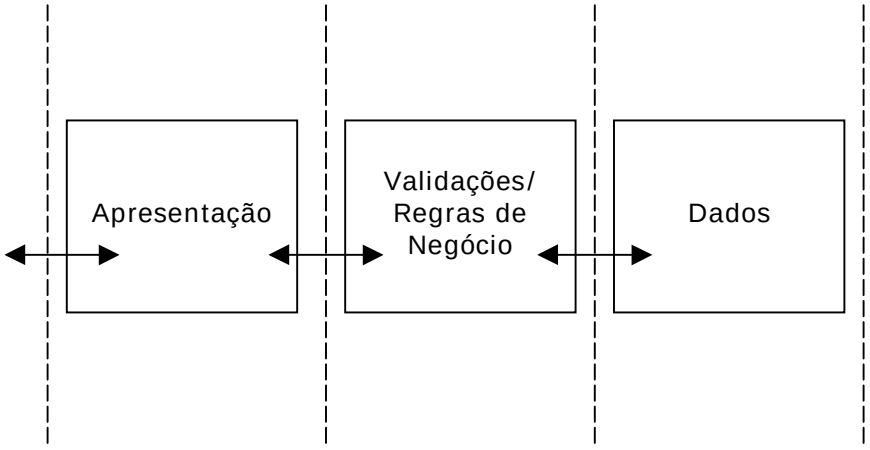

Figura 2 – Camadas de Software do Exemplo Modelo

[14] “Coupling”; “Cohesion” – Pressman, Roger. Software Engineering – A Practitioner’s Approach. 4th Edition. New York: McGraw-Hill, 1997. Pg. 357.

3.3.2 Determinar as Fronteiras

A fronteira indica os limites do software, determinando o que faz e o que não faz parte de cada unidade de software.

Do ponto de vista da implementação, as fronteiras são determinadas pela arquitetura. Dessa forma, a documentação da arquitetura precede a identificação das fronteiras. Em projetos de melhoria (manutenção evolutiva), as fronteiras devem ser traçadas com base em uma representação da arquitetura existente. Em projetos de novas aplicações, as fronteiras só poderão ser conhecidas após definida e documentada a arquitetura. No desenvolvimento incremental, diferentes iterações poderão resultar em modificações na arquitetura, as quais por sua vez poderão afetar o posicionamento das fronteiras.

3.3.2.1 Tipos de Fronteira

Existirá sempre uma fronteira separando as diferentes camadas de software. Esta é a "Fronteira entre Camadas".

Existirá também uma fronteira entre distintas unidades de software residentes na mesma camada ("Fronteira entre Pares" – *peers*).

Definido onde pode existir uma fronteira, vejamos como identificá-las.

3.3.2.2 Estímulos

Uma fronteira separa um estímulo do processo de software que trata esse estímulo. Um estímulo é comunicado à unidade de software através da entrada de dados, informações de controle, ou do passar do tempo. Um estímulo é uma representação de um evento do mundo real.

Exemplos comuns de estímulos são a entrada de dados ou informações de controle por parte do usuário. Outros exemplos são: a passagem de um intervalo de tempo, a chegada de uma informação esperada, o fim de um outro processo, um sinal de outro processo (trigger), etc.

Os estímulos são utilizados, em conjunto com as camadas, para estabelecer as fronteiras das unidades de software em uma determinada arquitetura.

3.3.2.3 Outras Considerações

Outros critérios utilizados na definição de fronteiras são coesão e acoplamento. Ver definições acima.

Unidades de software independentes são candidatas a serem desenvolvidas por fornecedores também independentes. A escolha de fornecedores independentes é facilitada pela existência de diferentes fronteiras.

Exemplo Modelo: Neste projeto de desenvolvimento, foram identificadas as seguintes fronteiras, ou unidades de software: Aplicação Web de Cadastramento de Clientes, Módulo de Validação de Clientes e Módulo de Atualização de Clientes. Ver Figura 3.

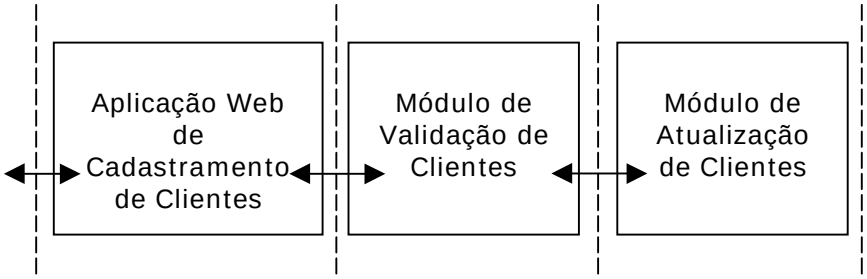

Figura 3 – Fronteiras do Exemplo Modelo

3.4 Contar Funções de Dados

A contagem de funções de dados nos PFI segue basicamente a mesma metodologia utilizada pelo IFPUG.

As funções de dados representam a funcionalidade provida ao usuário pela aplicação a fim de satisfazer requisitos de dados internos e externos. Os tipos de funções de dados são Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE).

Um Arquivo Lógico Interno (ALI) é um grupamento de dados logicamente relacionados, reconhecido pelo usuário e mantido por processos elementares do software considerado. Na contagem de PFI, estes processos elementares podem fazer uso de software contido em outras camadas ou fronteiras.

Um Arquivo de Interface Externa (AIE) é um grupamento de dados logicamente relacionados, reconhecido pelo usuário, referenciado para fins de recuperação de dados, mas não mantido por processo elementar da US considerada. Na contagem de PFI, o processo elementar pode fazer uso de software contido em outras camadas ou fronteiras. Um AIE deve, obrigatoriamente, ser mantido por processo elementar de alguma outra US, em outra fronteira.

A complexidade de um ALI ou AIE deve ser calculada observando-se as regras de contagem do IFPUG.

Para que os Pontos de Função referentes a um arquivo (ALI ou AIE) sejam contabilizados em uma US, é necessário que o arquivo seja diretamente acessado pela US. Caso o arquivo seja acessado apenas indiretamente pela US, serão contabilizados somente os PF correspondentes à referência ao mesmo (FTR[15]). Ver definição de acesso direto e indireto, acima.

Exemplo Modelo: Vamos verificar, para cada fronteira, quais os arquivos utilizados e candidatos a funções de dados na contagem de PFI. Assumiremos sempre o ponto de vista do usuário da contagem de PFI – um observador interno a cada fronteira considerada.

Aplicação Web de Cadastramento de Clientes – na visão do observador interno, esta unidade de software (US) irá atualizar o arquivo de Clientes. Para esta US, o arquivo de Clientes é um grupamento de dados logicamente relacionados, reconhecido pelo usuário e mantido por processo elementar a ela pertencente. No entanto, a atualização do arquivo de Clientes é

[15] A referência a um arquivo é denominada *File Type Reference*, ou FTR, na terminologia do IFPUG.

realizada de forma indireta por esta US, através de chamada a software residente em outra fronteira. Por este motivo, apenas a referência ao arquivo de Clientes (FTR) poderá ser contabilizada para os processos elementares desta US que o acessarem.

Módulo de Validação de Clientes – esta US irá acessar o arquivo de Clientes e verificar se os dados recebidos são válidos e se permitem a realização da operação desejada. Para este módulo, o arquivo de Clientes é um grupamento de dados logicamente relacionados, reconhecido pelo usuário e mantido por outra US. Porém, como o acesso ao arquivo de Cliente é realizado através de software não residente nesta US[16], apenas uma referência ao arquivo de Clientes poderá ser contabilizada para os processos elementares desta US que o acessarem.

Módulo de Atualização de Clientes – esta US gravará as informações recebidas no banco de dados de Clientes. Para esta US o arquivo de Clientes é um grupamento de dados logicamente relacionados, reconhecido pelo usuário e mantido por esta US. Como neste caso o acesso é direto, trata-se de um ALI para esta US. A Figura 4 ilustra esta situação.

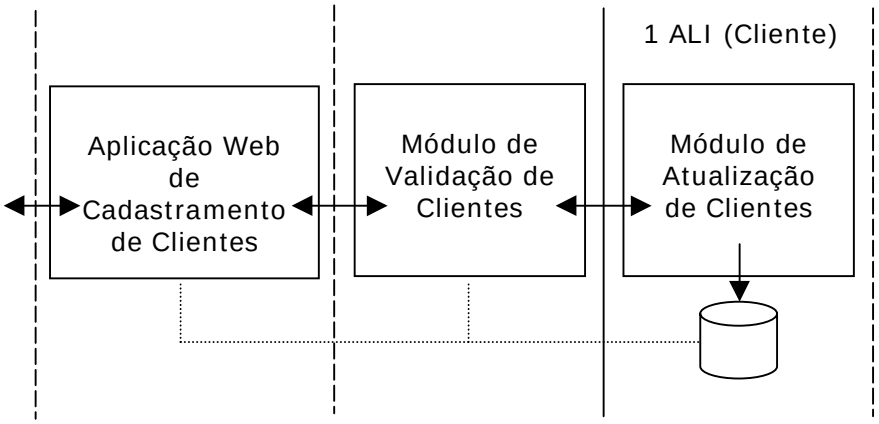

Figura 4 – Funções de Dados do Exemplo Modelo

A complexidade e contribuição correspondentes ao ALI e do AIE identificados seriam obtidas utilizando-se as regras de contagem de PF do IFPUG.

3.5 Contar Funções Transacionais

As funções transacionais utilizadas na contagem de PFI são as mesmas utilizadas na contagem de PF.

3.5.1 Entrada Externa

A intenção primária (ou objetivo principal) de uma Entrada Externa (EE) é atualizar os dados do sistema, ou modificar o seu comportamento. Uma EE processa dados ou informações de controle que vêm de fora da fronteira da US.

Atualizar os dados do sistema significa alterar os dados persistentes do mesmo, através de uma das seguintes operações: inclusão, alteração ou exclusão de dados.

[16] Acesso indireto – ver definição.

Modificar o comportamento do sistema é criar um efeito abrangente, iniciado através de um processo elementar e persistindo além do tempo de execução daquele processo. Um exemplo simples é a função "Escolher Impressora". Se um processo elementar altera a impressora escolhida, normalmente esta alteração afetará todos os demais processos da aplicação que tenham capacidade de imprimir, até que uma sessão seja encerrada, ou que outra impressora seja escolhida. Então, a ação "Escolher a Impressora" é considerada uma alteração no comportamento do sistema.

Na contagem de PFI, o principal elemento na identificação de funções transacionais é a intenção primária, que deve ser determinada a partir do ponto de vista do usuário "interno" à fronteira da unidade de software considerada.

Diferentemente do que é feito na contagem de PF, na contagem de PFI a atualização dos dados do sistema pode ser efetuada dentro ou fora da fronteira considerada, conforme seja direto ou indireto o respectivo acesso. Isto significa dizer que um ALI pode ser atualizado por uma US mesmo sem ser interno à sua fronteira. Ver exemplo adiante.

3.5.2 Saída Externa

A intenção primária (ou objetivo principal) de uma Saída Externa (SE) é apresentar informações ao usuário, efetuando processamento que não seja exclusivamente recuperação de dados ou informações de controle. Uma SE envia dados ou informações de controle para fora da fronteira da US. A lógica de processamento da SE deve conter alguma fórmula matemática ou cálculo, criar dados derivados, manter um ou mais ALI, ou alterar o comportamento do sistema.

Da mesma forma que no caso da EE, na contagem de PFI a recuperação ou atualização dos dados do sistema pode ser efetuada dentro ou fora da fronteira considerada.

3.5.3 Consulta Externa

A intenção primária (ou objetivo principal) de uma Consulta Externa (CE) é apresentar informações ao usuário, obrigatoriamente através da recuperação de dados ou informações de controle a partir de um ALI ou AIE. Uma CE envia dados ou informações de controle para fora da fronteira da US. A lógica de processamento da CE não pode conter fórmula matemática ou cálculo, criar dados derivados, manter ALI, ou alterar o comportamento do sistema.

Na contagem de PFI, a recuperação dos dados efetuada por uma CE pode ser efetuada dentro ou fora da fronteira considerada.

3.5.4 Exemplo da Identificação de Funções Transacionais

Exemplo Modelo: Identificar as funções de transação existentes nas fronteiras consideradas no projeto de desenvolvimento.

Aplicação Web de Cadastramento de Clientes – esta US apresenta ao usuário uma tela para cadastramento de clientes, permite que ele preencha os campos e clique OK. Caso os dados do cliente sejam validados corretamente, os mesmos são acrescentados ao banco de dados de Cliente.

Segundo a visão do usuário desta US, existe um processo elementar cuja intenção primária é atualizar os dados do sistema – cadastrar um cliente. O processo elementar recebe dados de fora da fronteira e, na visão do usuário (interno à fronteira da US), consulta (para validar) e atualiza o ALI de Cliente. Notar que, na contagem de PFI referente a funções transacionais,

não importa que a consulta e a atualização do ALI tenham sido implementadas em outra fronteira ou camada de software.

Temos então uma Entrada Externa (EE), cuja complexidade e contribuição devem ser obtidas de acordo com as regras de contagem de PF do IFPUG.

A consulta realizada ao ALI para validar os dados não é considerada um processo independente na visão deste usuário. A atividade de consulta é realizada somente como pré-requisito e em conjunto com a EE. Não é um processo em si, mas sim parte de um outro processo. Sua utilização isolada não faz sentido para o usuário desta fronteira.

Módulo de Validação de Clientes – esta US recebe um grupo de dados referentes a um cliente e efetua sua validação contra o registro correspondente da base de dados de Clientes. O usuário interno a esta fronteira reconhece a existência de um grupamento lógico de dados relacionados denominado "Clientes", que é mantido em outra US/fronteira. A intenção primária desta função transacional é apresentar informações ao usuário, a partir da recuperação de dados de um AIE. A lógica de processamento não envolve fórmula matemática ou cálculo, criar dados derivados, manter ALI, ou alterar o comportamento do sistema. Temos então uma Consulta Externa (CE).

Módulo de Atualização de Clientes – esta US recebe um grupo de dados referentes a um cliente e informações de controle que determinam a execução de uma inclusão, alteração ou exclusão na base de dados de Clientes. O usuário interno a esta fronteira reconhece a existência de um grupamento lógico de dados relacionados referentes a Clientes, que é mantido dentro desta fronteira. A intenção primária do processo elementar é atualizar os dados do sistema. Temos então uma Entrada Externa (EE).

3.5.5 Exemplo da Contribuição para a Contagem de PFI

Resumindo as considerações anteriores, a Figura 5 apresenta as funções de dados e transacionais creditadas a cada uma das fronteiras consideradas.

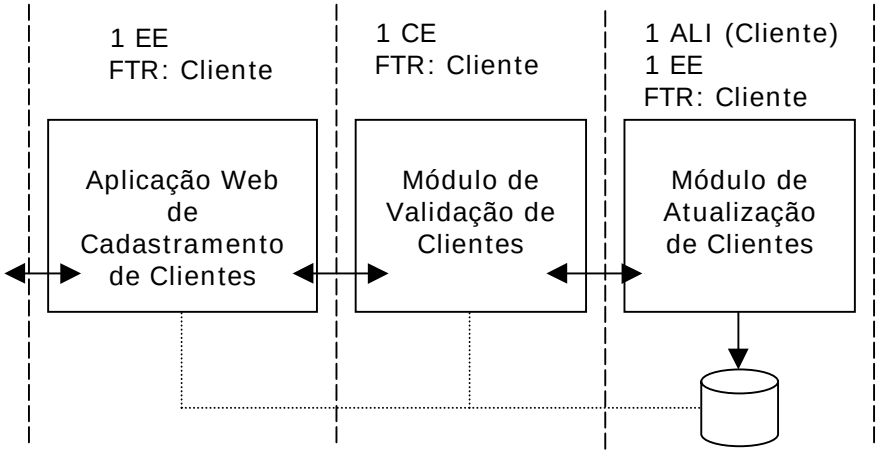

Figura 5 – Funções creditadas a cada US

3.6 Determinar PFI Não Ajustados

A contagem de PFI de uma aplicação composta por várias US é a soma das contagens de cada US. No Exemplo Modelo, supondo que todas as funções de dados tenham complexidade baixa e que todas as funções transacionais tenham complexidade média, teríamos os seguintes resultados:

Aplicação Web de Cadastramento de Clientes – 1 EE média = 4 PFI

Módulo de Validação de Clientes – 1 CE média = 4 PFI

Módulo de Atualização de Clientes – 1 ALI simples, 1 EE média = 7 + 4 = 11 PFI

Total de PFI Não Ajustados = 4 + 4 + 11 = 19 PFI

É interessante comparar este valor com a quantidade de PF correspondentes a esta aplicação. Se considerarmos todas as US dentro de uma só fronteira, teremos um ALI e uma única EE:

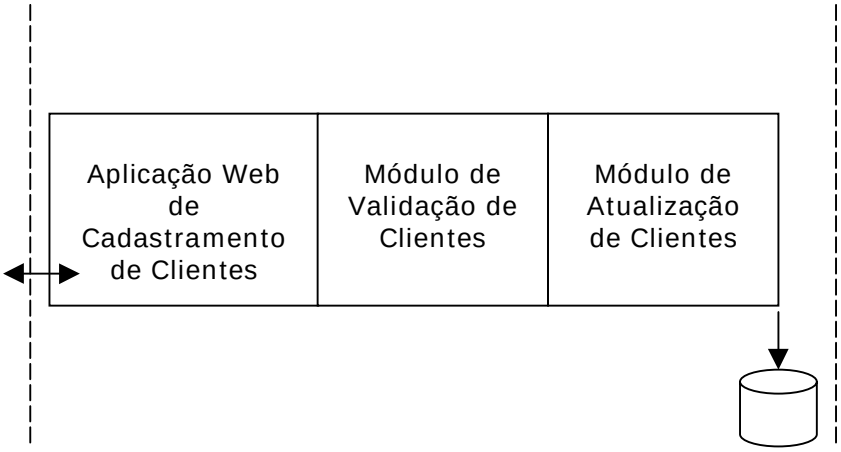

Figura 6 – Visão do Usuário do Negócio

A partir das premissas de complexidade anteriores, teremos aqui 11 PF: 7 Pontos de Função correspondentes a um ALI simples e 4 Pontos de Função correspondentes a uma EE média.

A comparação dos PF com os PFI poderá ser utilizada como uma medida da eficiência da implementação: o objetivo será implementar a mesma quantidade de PF com a menor quantidade possível de PFI. No caso, a razão PF/PFI = 11/19 é aproximadamente igual a 58%. Note-se que, para atribuir a este número um significado, teríamos que poder compará-lo a algum outro.

3.7 Determinar Fator de Ajuste

O Fator de Ajuste, se utilizado, deve ser determinado para cada US, consoante as regras do IFPUG. Notar que a utilização do Fator de Ajuste tornou-se opcional a partir da norma ISO/IEC 20929:2003.

3.8 Determinar PFI Ajustados

Após a determinação do Fator de Ajuste para cada US, deverá ser calculada a quantidade de PFI Ajustados para cada US. A quantidade de PFI Ajustados para a aplicação composta pelas US deverá ser obtida através da soma dos PFI Ajustados de todas as US envolvidas.

4 Conclusão

Os PFI – Pontos de Função de Implementação – são uma medida de tamanho do software obtida a partir da implementação. Trata-se, intencionalmente, de uma medida dependente da arquitetura e da implementação. Não medem o tamanho funcional dos requisitos, embora estejam indiretamente relacionados a eles.

Os PFI não são uma variação dos Pontos de Função do IFPUG. Ao contrário, constituem uma medida de software independente, que utiliza conceitos do IFPUG a fim de aproveitar a cultura existente.

Espera-se que esta primeira versão dos PFI – Pontos de Função de Implementação venha a ser objeto de críticas, sugestões, revisões, melhorias e ampliações, de modo a incorporar as perspectivas e contemplar as necessidades da comunidade brasileira usuária de métricas de software e, particularmente, da Análise de Pontos de Função.