COSMIC

João Carlos Testi Ferreira

Faculdade SENAI

Florianópolis, 2016

Sumário

O que é

A medição
funcional tem como característica
depender apenas dos requisitos
funcionais do usuário, o que torna
a medição independente de questões
técnicas, de solução ou qualidade.

Benefício

O fato de só depender
dos requisitos funcionais permite
que seja utilizada em conjunto com
outros métodos de esforço e favorece
a comparação entre projetos.

Origem do COSMIC

Em dezembro de 2002 o método COSMIC foi aceito como um padrão de medição de tamanho funcional com a norma ISO/IEC 19761.
Este método permite a medição de aplicações de negócio e aplicações de tempo real.

Médias

Na APF,
o máximo que uma transação pode medir é 7 PF. O que justifica isso?

Dados históricos

Estes números
vieram de dados históricos nos anos
de 1974 a 1979, com 22 projetos.

Suposições

Allan Albrecht arbitra os pesos:
4 para entradas, 5 para saídas,
4 para consultas e 10 para
arquivos mestre citando que
estes pesos apresentavam bons resultados de medida.

As estimativas

Usamos
medidas para fazer estimativas
para melhorar sua precisão.
Tratar distribuição de esforço
e custo nas funcionalidades quando
estas possuem teto é um problema.

O novo modelo

O COSMIC trata a medição pela movimentação de grupos de dados pelo sistema. Não há limites de tamanho para funcionalidade.

O armazenamento

Outro aspecto importante é que o COSMIC considera a movimentação da fronteira e a movimentação para o mecanismo de persistência.

Camadas

O COSMIC tem em
seu modelo o conceito de camada
e o papel de usuário do software
em uma camada superior em relação
ao software em uma camada inferior.

Porque modelo

Para medir o tamanho funcional de um pedaço de software é necessário mapear a definição de requisitos ou a implementação física a partir dos artefatos do pedaço de software para os conceitos dos modelos COSMIC. Estes conceitos são explicitados em seu modelo.

RFU

O COSMIC aplica um conjunto de modelos, princípios, regras e processos aos Requisitos Funcionais do Usuário (ou RFU) de um dado pedaço de software. O resultado é um número, o *valor de uma quantidade* representando o tamanho funcional do pedaço de software de acordo com o método COSMIC em unidades de *Pontos de Função COSMIC* (ou PFC).

Característica dos RFU

RFU pode ser incompleto
ou apresentar informações a serem abstraídas. Nem sempre os RFU existem, e suas informações podem ser obtidas por engenharia reversa.

Princípios

O modelo de contexto
do software apresenta princípios
e conceitos para sua definição.

Princípios (1/3)

a O Software é delimitado pelo hardware

b O software é normalmente estruturado em camadas

c Uma camada pode conter um ou mais *pares* de pedaços de software distintos e qualquer pedaço de software pode ser composto por componentes pares distintos

d Qualquer pedaço de software a ser medido deverá ser definido por seu escopo de medição, o qual deve estar integralmente contido em uma única camada

e O escopo de um pedaço de software a ser medido depende do propósito da medição

Princípios (2/3)

f Os usuários funcionais de um pedaço de software devem ser identificados a partir dos requisitos funcionais do usuário do pedaço de software a ser medido, como fontes e/ou destinos pretendidos para os dados

g Um pedaço de software interage com os seus usuários funcionais por meio de movimentações de dados através de uma fronteira, e o pedaço de software pode mover dados de e para o armazenamento persistente dentro da fronteira

h Os RFU do software podem ser expressos em diferentes níveis de granularidade

Princípios (3/3)

i O nível de granularidade no qual as medições devem ser normalmente efetuadas é o dos processos funcionais

j Se não for possível medir no nível de granularidade dos processos funcionais, nesse caso os RFU do software devem ser medidos através de uma abordagem de aproximação e escalonados para o nível de granularidade dos processos funcionais

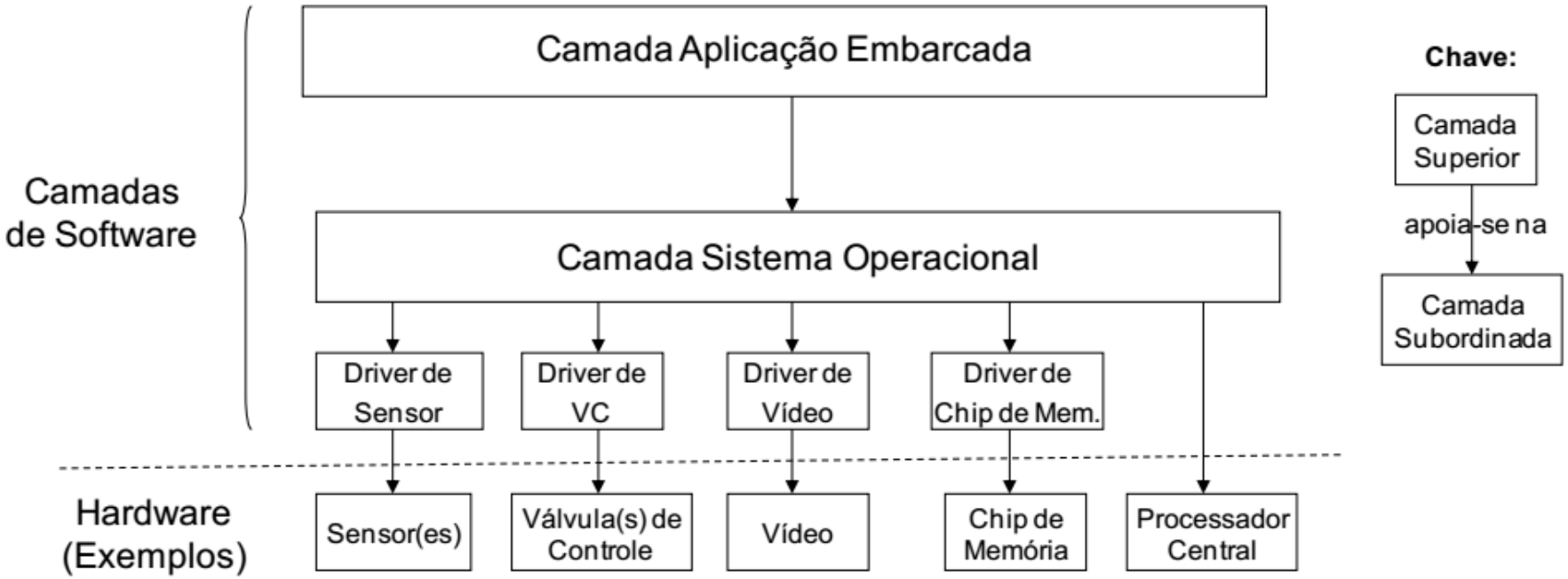
Visão física

Para poder detalhar os princípios é preciso compreender as visões computacionais.

Visão Física

Mostra como o software é normalmente estruturado em uma hierarquia de camadas, cada uma com sua especialidade própria. Esta visão mostra que, na realidade, toda comunicação com qualquer pedaço de software acontece via dispositivos de hardware e (talvez) outras camadas intermediárias de software.

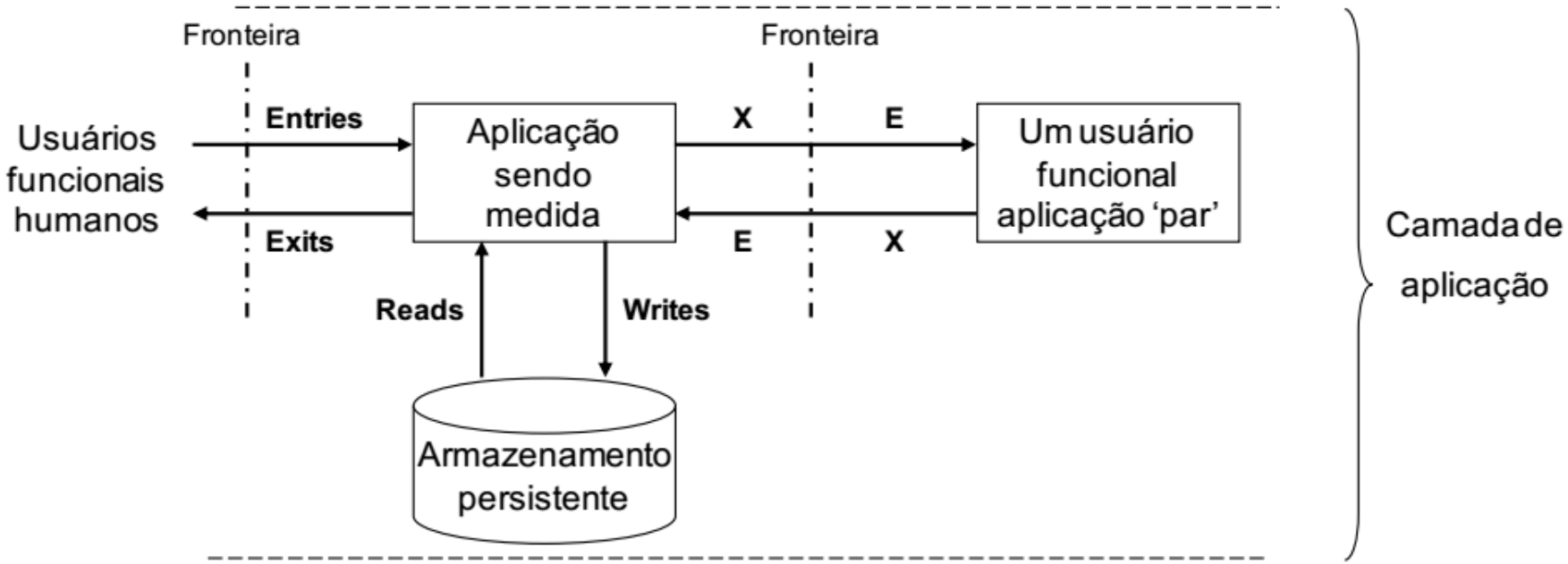
Visão física

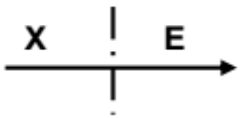


Visão lógica

Visão Lógica
Uma abstração da visão física utilizada para fins de medição do tamanho funcional. Esta visão mostra que os usuários funcionais de um pedaço de software a ser medido (fontes e/ou destinos pretendidos para os dados) interagem com o software através de uma fronteira e que o software movimenta dados de e para o armazenamento persistente. Neste nível de abstração todo hardware e software que permitem tais interações são ignorados.

Visão lógica





Indica uma mensagem enviada como uma movimentação de dados Exit que atravessa a fronteira e é recebida como uma movimentação de dados Entry

Princípio a

O software usado por seres humanos é delimitado por dispositivos de hardware de entrada/saída tais como: mouse, teclado, impressora ou monitor; o software *real-time* geralmente é delimitado por dispositivos tais como sensores ou relés. O software também é delimitado por hardware de armazenamento de dados tal como disco rígido, ou outros tipos de memória utilizados para guardar dados.

Princípio b

Se o pedaço de software a ser medido é parte de uma arquitetura desenhada em camadas, deve ser fácil decidir à qual camada o pedaço pertence. Entretanto, se um ambiente de software cresceu e evoluiu ao longo do tempo, as camadas (se houver) não poderão ser claramente identificáveis. Para estas situações, o método COSMIC inclui algumas regras para identificar camadas.

Princípio c

Os principais componentes distintos de uma aplicação de negócios (por exemplo, uma arquitetura ‘três-camadas’ com um componente de ‘interface com o usuário / apresentação’, um componente de ‘regras de negócios’ e um componente ‘serviços de dados’) são componentes pares. Tais componentes principais podem ser medidos separadamente e o método COSMIC fornece as regras para tal dimensionamento. Esta habilidade de medir separadamente os tamanhos dos principais componentes de um pedaço de software quando os mesmos executam em plataformas técnicas diferentes é muito importante, na prática, para os próposito de medição de performance e estimativas.

Princípio d

O escopo de qualquer pedaço de software medido deva estar inteiramente contido em uma única camada. A razão disto é que cada camada tem uma função especializada e pode ser desenvolvida com tecnologia diferente daquela utilizada nas outras camadas. Dessa forma, pode ou não fazer sentido medir o tamanho de alguns pedaços de software residentes em duas ou mais camadas e, depois, somar tais tamanhos como se o resultado representasse o tamanho de uma simples entidade. A medida de tamanho resultante poderia, assim como a soma dos tamanhos de maçãs e laranjas, ser muito difícil de interpretar e/ou comparar com outras medidas de tamanho funcional. Para as regras de agregação do tamanho de pedaços de software em diferentes camadas.

Princípio e

Suponha que os pedaços da aplicação de software apresentado na visão física sejam cada um pedaços distintos de software, desenvolvidos por equipes de projeto distintas. Para a maioria dos propósito de medição mais comuns, faria sentido definir o escopo da medição como a *aplicação como um todo*. Mas, se uma aplicação for desenvolvida como três componentes pares principais, cada um utilizando tecnologias diferentes e/ou desenvolvidos por diferentes equipes de projeto, então para o propósito de estimar o esforço do projeto fará sentido definir três escopos de medição separados, um para cada componente par.

Princípio f

A visão lógica mostra a interação dos *usuários funcionais*, com um pedaço de software de aplicação de negócio. Poderiam ser considerados *usuários* da aplicação o sistema operacional, qualquer um dos dispositivos de hardware (por exemplo, o teclado, a impressora, etc.), e os usuários humanos, uma vez que todos eles *interagem* direta ou indiretamente com a aplicação. Para um pedaço de software de uma aplicação de negócio, em geral os RFU descrevem somente a funcionalidade requerida do ponto de vista dos usuários humanos da aplicação e, talvez, outras aplicações pares que enviem ou recebam dados de/para a aplicação. Tais seres humanos e aplicações pares serão, portanto, os usuários funcionais da aplicação.

Princípio g

A visão lógica mostra que os usuários funcionais interagem com o software através da fronteira via dois tipos de movimentação de dados (Entries e Exits). Também troca dados com o dispositivo de armazenamento persistente via dois tipos de movimentação de dados (Reads e Writes). A *fronteira* é definida como *uma interface conceitual entre o software e seus usuários funcionais*. A fronteira permite fazer uma distinção clara entre qualquer coisa que seja parte do pedaço de software sendo medido (dentro da fronteira do software) e qualquer coisa que seja parte do ambiente dos usuários funcionais (fora da fronteira do software). O dispositivo de armazenamento não é considerado como um usuário do software e portanto está dentro da fronteira do software.

Princípios h e i

O RFU pode ser expresso em diferentes *níveis de granularidade*. O nível de granularidade no qual a medição deve ser realizada é o do processo funcional. Ao iniciar um novo desenvolvimento de software, o processo de determinação dos RFU do software normalmente começa com a definição e concordância a respeito de uma declaração de requisitos de *alto nível*, os quais são refinados e trabalhados em maior detalhe. Os RFU de um pedaço de software de um dado escopo podem dessa forma existir em diferentes níveis de granularidade. Na medida que detalhamos o RFU o tamanho funcional parece aumentar, pois mais detalhes são levados em consideração. Medidas com RFU em níveis de granularidade diferentes são como medidas em escalas.

Princípio j

O problema de se precisar medir um pedaço de software em um nível de granularidade mais alto que o de processo funcional geralmente aparece apenas nas primeiras fases do desenvolvimento de um novo software, enquanto os requisitos ainda estão em evolução. Em tais circunstâncias, quando não é possível medir no nível de granularidade dos processos funcionais, uma abordagem de medição por aproximação deve ser utilizada e o resultado escalonado para o nível de granularidade dos processos funcionais.

Porque princípios

Da
mesma forma que temos princípios
para a interpretação dos requisitos,
temos princípios para interpretar
um modelo que atenda qualquer
tipo de software - modelo genérico.

Princípios (1/3)

a O software recebe dados de entrada dos seus usuários funcionais, gerando saídas e/ou outros resultados para os usuários funcionais

b Os requisitos dos usuários funcionais de um pedaço de software a ser medido podem ser mapeados para processos funcionais distintos

c Cada processo funcional consiste de subprocessos

d Um subprocesso pode ser uma movimentação de dados ou uma manipulação de dados

e Cada processo funcional é disparado por um movimento de dados do tipo Entry proveniente de um usuário funcional, o qual informa ao processo funcional que o usuário funcional identificou um evento

Princípios (2/3)

f Uma movimentação de dados movimenta um único grupo de dados

g Um grupo de dados consiste de um único conjunto de atributos de dados que descreve um único objeto de interesse

h Há quatro tipos de movimentação de dados. Uma Entry movimenta um grupo de dados para dentro do software, a partir de um usuário funcional. Uma Exit movimenta um grupo de dados para fora do software, em direção a um usuário funcional. Um Write movimenta um grupo de dados do software para o armazenamento persistente. Um Read movimenta um grupo de dados do armazenamento persistente para o software

Princípios (3/3)

i Um processo funcional deve incluir pelo menos uma movimentação de dados Entry e uma movimentação de dados Write ou Exit, ou seja, deve incluir no mínimo duas movimentações de dados

j Como uma aproximação para fins de medição, os subprocessos de manipulação de dados não são medidos separadamente; assume-se que a funcionalidade de qualquer manipulação de dados será considerada na movimentação de dados com a qual a mesma esteja associada

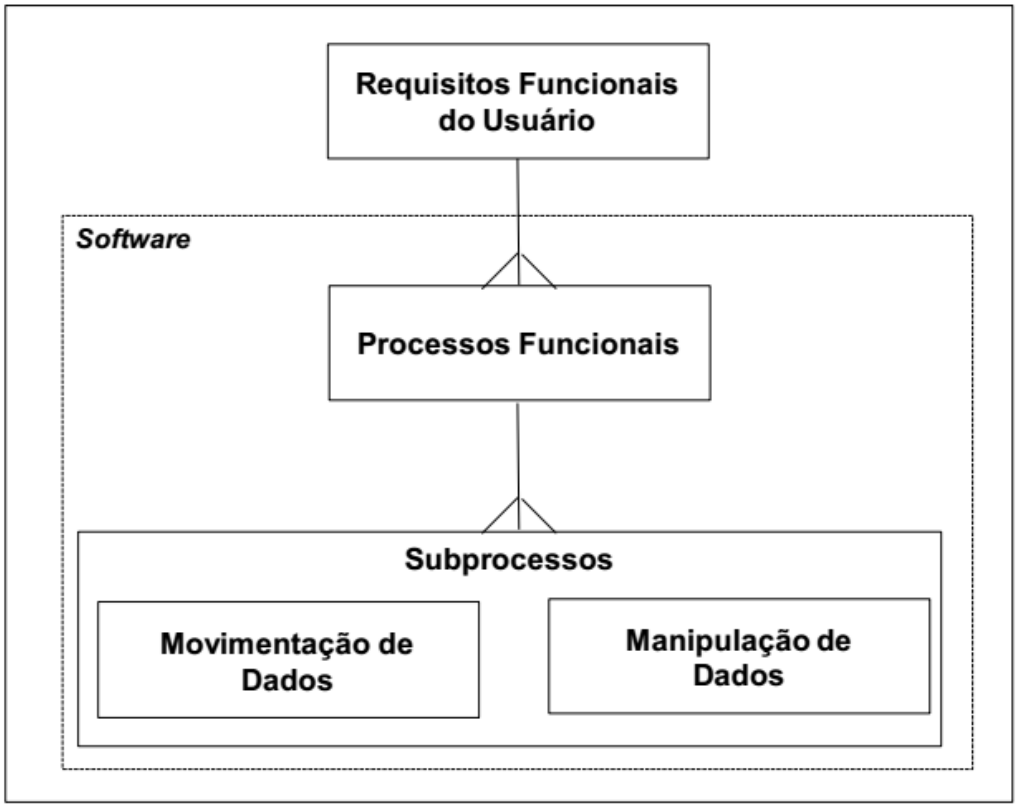
Princípios a, b

A tarefa do software é responder a eventos que ocorrem do lado de fora da sua fronteira, isto é, no mundo dos usuários funcionais. Um usuário funcional notifica o software da ocorrência de um evento e pode enviar dados relacionados ao evento. O software deve fazer alguma coisa útil para o usuário funcional em resposta a aquele evento. Isso é o *processo funcional*. Todos os RFU de software portanto podem ser expressos como uma lista de tipos de eventos e processos funcionais correspondentes que executam a resposta do software para cada evento.

Princípios c, d

Os processos funcionais podem ser compostos de dois tipos de subprocessos, denominados movimentação de dados e manipulação de dados. O software somente pode movimentar e/ou manipular dados.

Princípios b e d



Princípios f, g (1/4)

Cada movimentação de dados transporta apenas um grupo de dados, isto é, dados sobre um simples objeto de interesse, ou seja, algo *de interesse* para o usuário funcional.

Princípios f, g (2/4)

Por exemplo, no domínio de software de aplicação de negócio, um processo funcional relativamente simples para registrar um pedido tipicamente envolve a seguinte movimentação de dados (objetos de interesse são apresentados em itálico):

- Duas Entries de grupos de dados relacionados ao *pedido* e ao *item do pedido* (assumindo um pedido com vários itens). A primeira destas Entries de dados descrevendo o objeto de interesse *pedido* é uma das que dispara (ou inicia) o processo funcional
- Dois Reads de grupos de dados relacionados a *cliente* e *produto*, para validar que o cliente está autorizado a fazer o pedido e que o produto solicitado é valido e está disponível.

Princípios f, g (3/4)

- Writes de grupos de dados relacionados ao *pedido* e ao *item do pedido*, para movimentar os dados informados para o armazenamento persistente
- Uma ou mais Exits de grupos de dados contendo, por exemplo, uma *confirmação do pedido*, uma mensagem incluindo o valor total do pedido, uma orientação ao almoxarifado para separar cada *item do pedido*, etc.

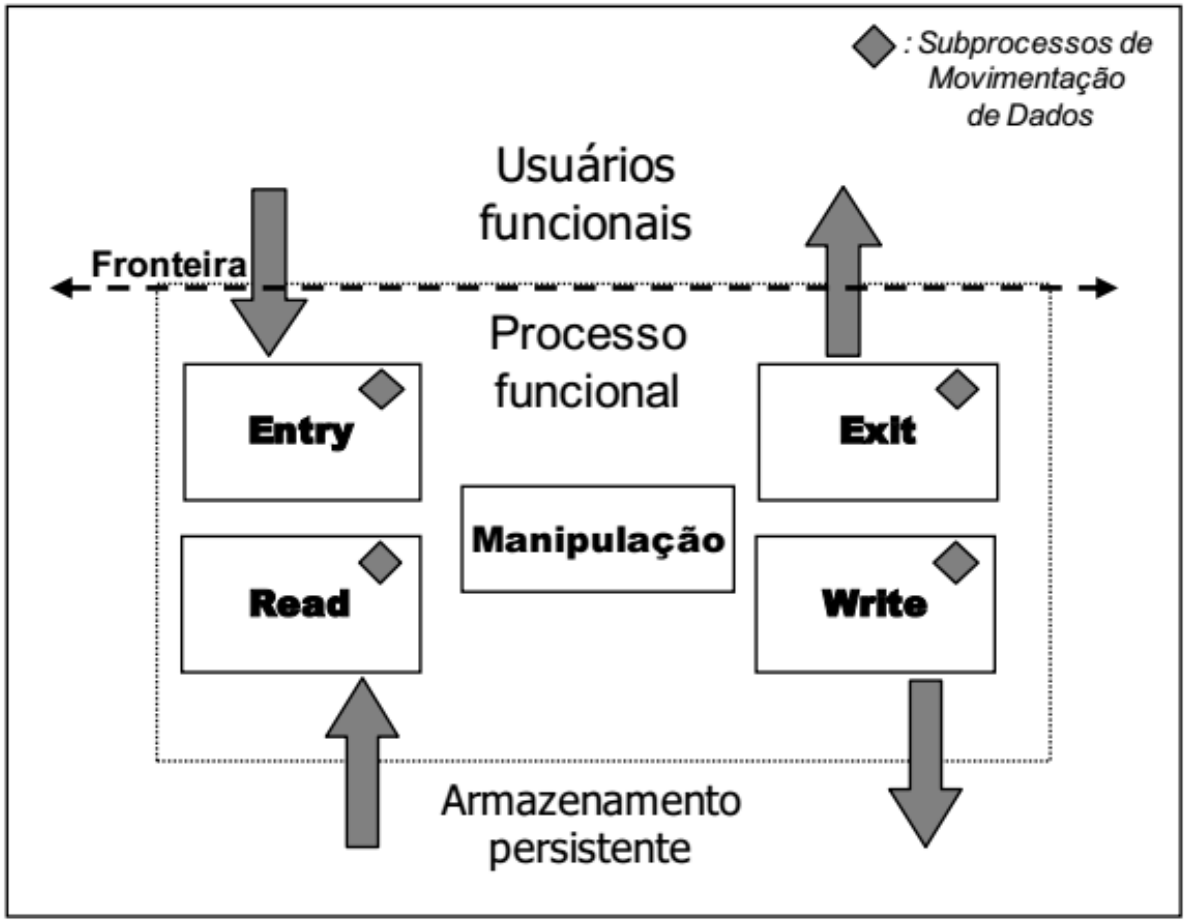
Princípios f, g (4/4)

Todos estes objetos de interesse são coisas reais ou conceituais no mundo real dos usuários funcionais (humanos, neste caso), sobre as quais o pedaço de software é solicitado a processar dados. Devem ser identificados e distinguidos de forma a identificar as movimentações de dados.

Princípio h

Este princípio afirma que os quatro tipos de movimentação de dados são distinguidos por sua origem e destino. As movimentações de dados cruzam a fronteira entre o software sendo medido e seus usuários funcionais (Entries e Exits), ou movimentam-se entre o software e o armazenamento persistente (Reads e Writes).

Princípio h


: Subprocessos de Movimentação de Dados
Usuários funcionais
Fronteira
Processo funcional
Entry
Exit
Manipulação
Read
Write
Armazenamento persistente

Princípio i

Este princípio afirma que o processo funcional deve ter pelo menos duas movimentações de dados. Isto decorre dos princípios anteriores. Um processo funcional que apenas recebesse uma movimentação de dados e nada fizesse seria praticamente inútil. Portanto, todos os processos funcionais devem ter pelo menos uma movimentação de dados informando sobre a ocorrência de um evento (uma Entry), e pelo menos uma outra movimentação de dados como resposta (ou resultado útil), para um usuário funcional (uma Exit) ou para o armazenamento persistente (um Write).

Princípio j (1/2)

Para fins de medição, e dado o domínio de software para o qual o método foi desenhado, o método COSMIC assume uma simplificação do Modelo Genérico de Software. Como uma primeira aproximação nesta versão do método de medição, subprocessos do tipo manipulação de dados não são reconhecidos separadamente, e sim considerados associados com ou parte de um subprocesso de movimentação de dados.

Princípio j (2/2)

Dada esta aproximação, podemos ver por que o método padrão COSMIC é adequado para medir tipos de software *ricos em movimentações*, tais como a maioria das aplicações de negócio e muitos softwares de tempo real, porém inadequado para medir software *rico em manipulação* (ou *rico em algoritmos*). O Manual de Medição salienta também a necessidade de cuidado ao medir pedaços de software muito pequenos, especialmente pequenas mudanças de software, onde a hipótese do princípio (j) pode deixar de ser válida.

RFU

Os requisitos funcionais de usuário são os que descrevem o que o software deve fazer, como tarefas ou serviços:

- transferir dados: informar dados de cadastro de cliente
- transformar dados: calcular os juros de um determinado empréstimo
- armazenar dados: persistir os dados de cadastro do cliente
- recuperar dados: listar os empregados cadastrados.

RNF

Os requisitos não funcionais geralmente são restrições aplicadas a todo o sistema de software/hardware ou ao projeto de desenvolvimento.
Muitas vezes RNF identificados nas fases iniciais do projeto evoluem para RFU durante o detalhamento dos requisitos.

Exemplo

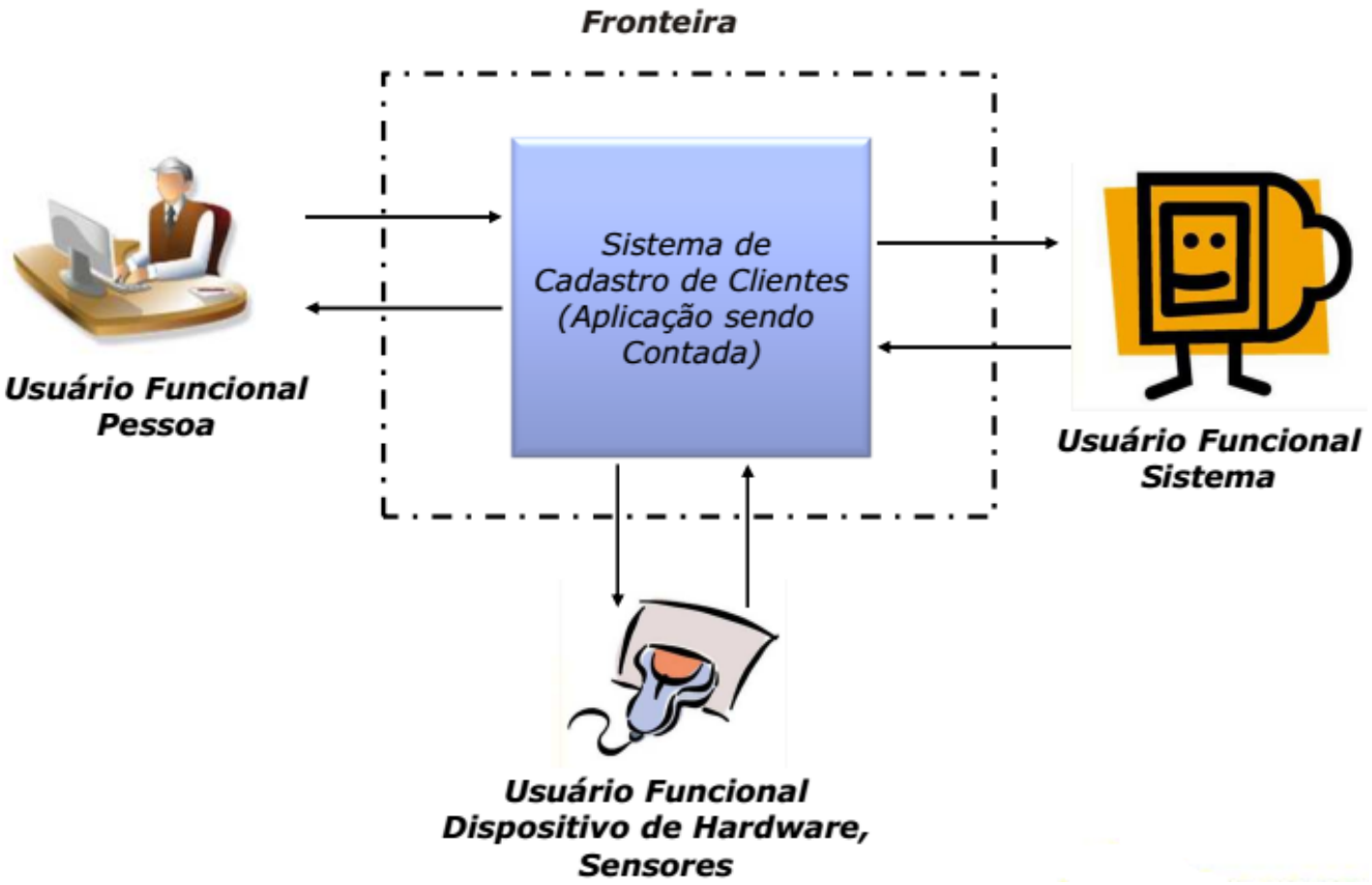
Usando um sistema de gestão de pessoas como exemplo, temos os requisitos:

- RFU – a manutenção de dados de empregados da empresa como por exemplo: nome, endereço, cargo, departamento, qualificações, etc.
- RNF – controle de acesso, disponibilidade do sistema, tecnologia de desenvolvimento, data limite de entrega.

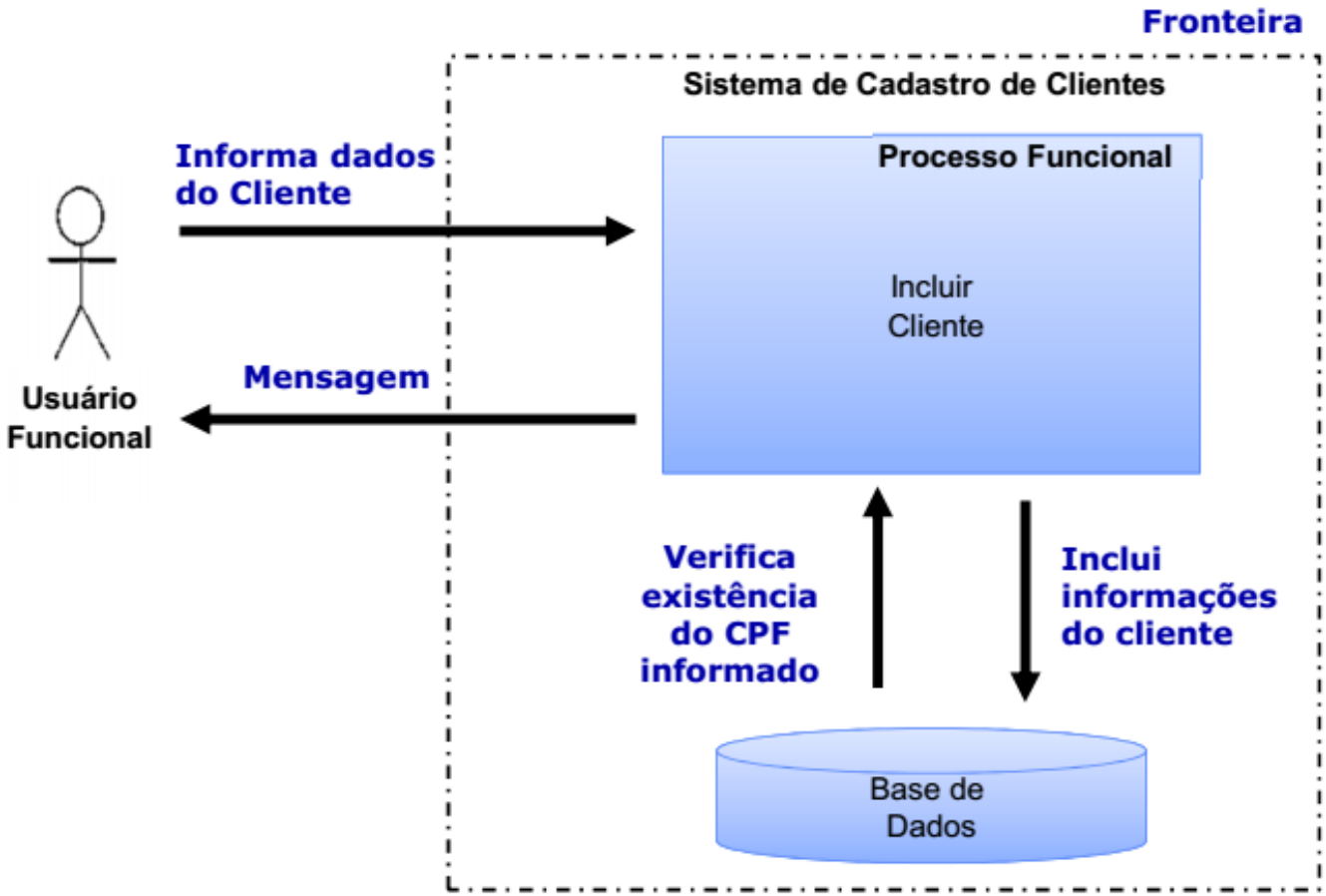
Movimentação de dados

A movimentação de dados envolve o que o usuário informa ao sistema, recebe do sistema e os dados que o sistema persiste e recupera.
Cada movimentação de dados é contada como um ponto (1 PFC).
O tamanho funcional será a soma dos pontos obtidos pelas movimentações de dados.

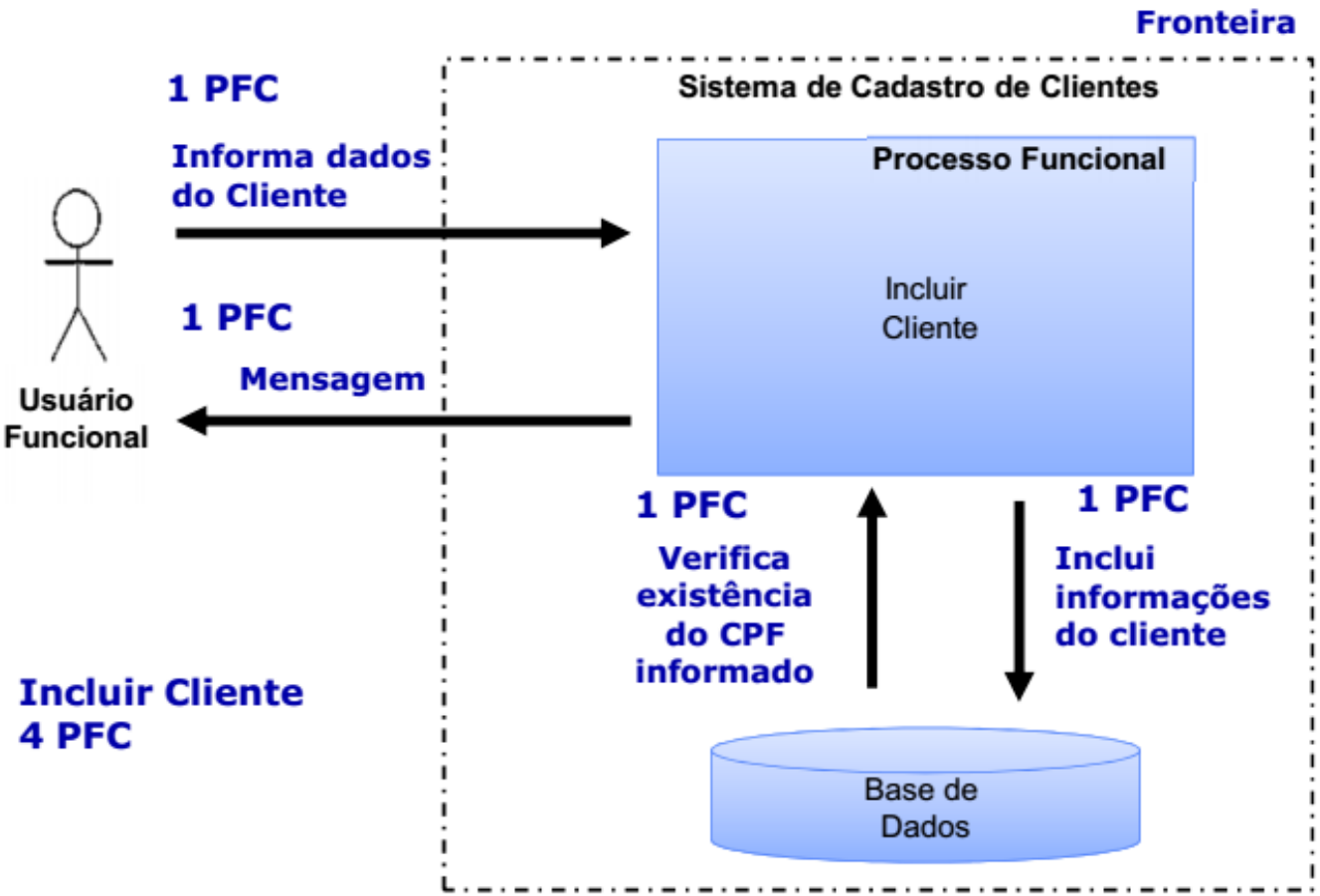
Elementos de contagem



Elementos de contagem



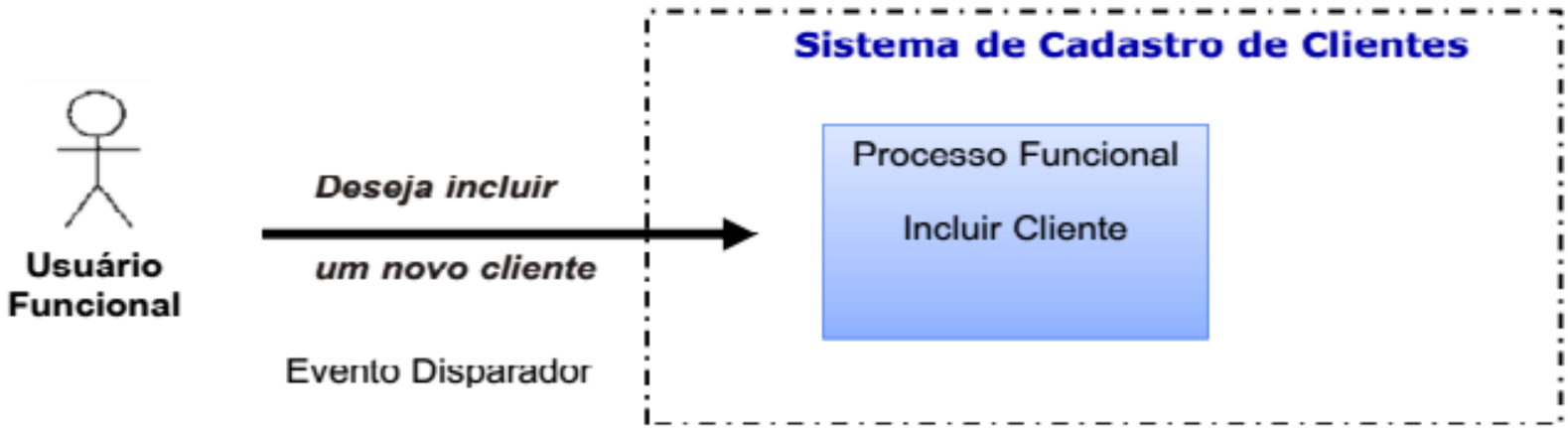
Contagem


Fronteira
Sistema de Cadastro de Clientes
1 PFC
Informa dados do Cliente
Processo Funcional
Incluir Cliente
1 PFC
Mensagem
Usuário Funcional
1 PFC
Verifica existência do CPF informado
1 PFC
Inclui informações do cliente
Incluir Cliente
4 PFC
Base de Dados

Processo funcional

É um conjunto de movimentações de dados que representa uma parte dos Requisitos Funcionais do Usuário para o pedaço de software sendo medido. O conjunto de movimentações de dados é único, coeso e independentemente executável. O conjunto de movimentações de dados é aquele necessário para atender todas as possíveis respostas ao evento disparador. Ele inicia quando um usuário funcional percebe um evento e aciona o processo, e encerra quando tiver executado tudo o que é requerido em resposta ao evento disparador.

Identificação do processo funcional



Formas de medir

A medição pode ser objetiva ou não. As estimativas, de mesma forma, podem basear-se na experiência ou em algoritmos que permitem fazer previsões. A diversidade de formas de medir causa problemas por não podermos compara-las.

Medidas objetivas

Temos várias formas de medir objetivamente e as mais populares são medidas funcionais. Medidas funcionais permitem produzir estimativas e indicadores que auxiliam no processo de desenvolvimento, em especial em sua melhoria. A grande vantagem das medidas funcionais é que não levam em consideração a tecnologia empregada, baseando-se na perspectiva do usuário. Isso da estabilidade à medição.

Medidas objetivas

O ágil normalmente usas medidas subjetivas, baseada em conhecimento para poder planejar. Essas medidas, da mesma forma que as anteriores permitem que sejam estimados e avaliados aspectos de melhoria, contudo, acabam sendo dependentes de tecnologia e do conhecimento da equipe. São difíceis de comparar.

Estimativas

Há formas de medir conforme a quantidade de informação que temos disponível. A precisão aumenta na medida que mais informações possuímos, mas mesmo com pouca informação é possível fazer estimativas de tamanho que permitam seu uso para planejamento.

Para saber mais ...


PRESSMAN, R. S. *Engenharia de Software*: Uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. ISBN 9788580550443.


SOMMERVILLE, I. *Engenharia de Software*: Uma abordagem profissional. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 9788579361081.


VASQUEZ, C. E.; SIMõES, G. S.; ALBERT, R. M. *Análise de Pontos de Função*: Medição, estimativas e gerenciamento de projetos de software. 6. ed. São Paulo: Editora Érica, 2007. ISBN 9788571948990.


ALEXANDER, J. *Codermetrics*: Analytics for improving software teams. Sebastopol: O'Reilly Media, 2011. ISBN 9781449305154.


FAIRLEY, R. E. *Managing and Leading Software Projects*. New Jersey: John Wiley and Sons, 2009. ISBN 9780470294550.


RASMUSSON, J. *The Agile Samurai*: How agile masters deliver great software. Dallas: Pragmatic Bookshelf, 2010. ISBN 9781934356586.