JOURNAL 3

JOURNAL3 JORNAL DE ARQUITETOS DA MICROSOFT JULHO DE 2004 A NOVA PUBLICAÇÃO PARA ARQUITETOS DE SOFTWARE

Caro Arquiteto

Embora freqüentemente seja complicado entrar em acordo, algo que eu penso em que todos os arquitetos deveriam concordar é que nosso trabalho está cada vez mais difícil, e não mais fácil. Estamos enfrentando um nível de complexidade que tende sempre a aumentar, com recursos sempre em diminuição. Sejam novos desafios, como o aumento da conformidade com as regulamentações, ou desafios antigos, como problemas com recursos ou orçamentos cortados, o trabalho do arquiteto continua a ficar mais difícil e mais importante a cada dia.

A abstração é a principal ferramenta do arquiteto para lidar com a complexidade. Já vimos a evolução das técnicas de abstração arquitetônica, como modelos e padrões; no entanto, ainda temos que criar muito valor, mensurável, a partir delas. Modelos e padrões são úteis para nos comunicarmos de forma eficaz com nossos colegas; porém, até o momento, não ajudaram a reduzir drasticamente a quantidade de recursos necessários para criar e operar sistemas. Para continuar a lidar com a complexidade em crescimento, precisamos nos tornar muito mais pragmáticos com relação ao uso da abstração para solucionar problemas.

Nesta publicação, a terceira edição do JOURNAL, nos concentramos no aumento do nível de abstração. Esse enfoque vai das técnicas gerais para abstração até a introdução de uma abordagem de fabricação industrializada para o desenvolvimento de software. Pelo caminho, iremos abordar o uso de modelos para descrever funções de negócio, bem como o gerenciamento de identidade e acesso.

Independente de nossas discordâncias, a crescente crise de complexidade ameaça impedir o progresso da construção de software e os negócios que dependem desses softwares. Em suas edições futuras, o JOURNAL irá continuar a oferecer informações sobre todos os aspectos da indústria de arquitetura de software e a fornecer a orientação de que os arquitetos precisam.

Harry Pierson
arquiteto
Estratégia de arquitetura da D&PE
Microsoft Corporation

Editorial

Por Arvindra Sehmi

Caro Arquiteto

Bem-vindo à nova edição do JOURNAL. Estes últimos meses foram muito interessantes para a "arquitetura" como um tópico na Microsoft. O centro de arquitetura da Microsoft® se estabeleceu como o portal líder para conteúdo arquitetônico e como um trampolim para que milhares de nossos clientes e parceiros pudessem mergulhar na excelente orientação em arquitetura e desenvolvimento de soluções na plataforma Windows®.

Esta edição está repleta de preciosidades da arquitetura escritas por arquitetos renomados da Microsoft e por parceiros de valor, por isso estou confiante de que a qualidade do conteúdo subiu a um novo patamar.

Iniciaremos com um trabalho escrito por Don Awalt e Rick McUmber da RDA Corporation, que também são membros do Comitê de Consultoria em Arquitetura da Microsoft e revelam vários segredos de grandes arquitetos. Eles abordam os problemas enfrentados diariamente por arquitetos empresariais, especialmente o desafio da alta complexidade em desenvolvimento de sistemas que é exacerbada pelas necessidades constantes de alteração dos negócios e pela pressão em se adotar novas tecnologias assim que surgem. O principal segredo dos grandes arquitetos aqui revelado começa com o domínio da conceituação e da abstração de soluções. A forma como os autores dissecam o problema e fornecem uma revisão exemplar do processo de solução é a própria evidência de tal domínio.

Em seu artigo, Jack Greenfield, da divisão de ferramentas e estruturas empresariais da Microsoft, discute novas idéias sobre um sério problema que preocupa várias organizações na atualidade – como melhorar o desenvolvimento de software? Na prática atual, o desenvolvimento de software é lento, caro e propenso a erros, e resulta em uma variedade de problemas já conhecidos. Apesar dessas deficiências, os "produtos" de desenvolvimento de software obviamente fornecem valor significativo aos consumidores, conforme mostrado pela tendência, em longo prazo, de um aumento de demanda. Para solucionar essas falhas, descreve-se um caso para a metodologia "Fábricas de Software" que se propõe a mostrar técnicas que permitam industrializar o desenvolvimento de software, descrito em detalhes no livro de mesmo nome de Jack Greenfield e Keith Short, da John Wiley and Sons.

Os feedbacks de clientes para a Microsoft sobre os desafios de implementar sistemas de SOA foram muito consistentes; problemas em gerenciar identidades, agregar dados, gerenciar serviços e integrar processos de negócio foram citados inúmeras vezes como os principais obstáculos para criar organizações mais ágeis e eficientes. Frederick Chong da equipe de estratégia de arquitetura na Microsoft escreveu um artigo sobre um desses desafios, o Gerenciamento de Identidade e Acesso. Ele fornece uma visão geral sucinta e abrangente do que o Gerenciamento de Identidade e Acesso representa, utilizando uma estrutura simples que consiste em três áreas principais: gerenciamento do ciclo de vida da identidade, gerenciamento do acesso e serviços de diretório.

Philip Teale da Microsoft e Robert Jarvis da SA Ltd. continuam com a segunda parte do artigo que discute business patterns – que são, essencialmente, modelos arquitetônicos para soluções de negócio. Nesse artigo, eles descrevem como desenvolver business patterns baseados em funções, dados e componentes de negócio e também mostram como esses elementos podem ser usados na engenharia de sistemas de software. Um exemplo real, porém simplificado, é usado para mostrar como utilizar técnicas padrão para desenvolver descrições desses elementos necessários para algum business patterns.

Em seguida, Easwaran Nadhan e Jay-Louise Weldon, ambos da EDS, examinam várias estratégias de transferência de dados, que permitem acesso oportuno às informações corretas e compartilhamento de dados de forma que os processos de negócio possam ser mais eficientes em toda a empresa. Eles descrevem oito opções e as analisam utilizando critérios, como requisitos de latência de dados, necessidades de transformação, considerações sobre volume de dados, restrições relacionadas ao nível de invasão e esforço tolerado pela empresa para poder obter os benefícios esperados.

O último artigo é a segunda parte da descrição de Soumen Chatterjee sobre os messaging patterns de SOA. Tradicionalmente, messaging patterns foram aplicados a soluções de integração de aplicações empresariais. Soumen usa esses patterns para explicar como a SOA pode ser implementada. Seus insights derivam-se do trabalho original do livro de Hohpe e Woolf sobre Enterprise Integration Patterns. No entanto, Soumen nos mostra como os mesmos messaging patterns descritos no livro podem ser aplicados com a mesma eficácia às arquiteturas de aplicações, especialmente em soluções baseadas em SOA, porque também são fundamentalmente orientadas a mensagens.

Mantenha-se atualizado no centro de arquitetura da Microsoft® e especificamente na página inicial do JOURNAL, http://msdn.microsoft.com/architecture/journal, onde será possível baixar os artigos para sua comodidade. Além disso, fique atento aos comunicados do novo serviço de inscrição do JOURNAL, que estará disponível em breve.

Como sempre, caso esteja interessado em escrever para o JOURNAL, envie um breve resumo de seu tópico e seu currículo para asehmi@microsoft.com.

Agora descanse e aproveite a leitura!

Arvindra Sehmi
Arquiteto, D&PE, Microsoft EMEA

Segredos de Grandes Arquitetos

Por Don Awalt e Rick McUmber, RDA Corporation

Aplicando Níveis de Abstração a Soluções de TI
Aplicando Níveis de Abstração a Soluções de TI

Arquitetos empresariais são desafiados pela vasta quantidade de complexidade que enfrentam. Uma coisa é desenvolver uma aplicação departamental isolada que automatiza alguma tarefa de negócio. Outra, bem diferente, é criar e montar uma rede internacional de laboratórios de TI repleta de aplicações, servidores e bancos de dados que oferecem suporte a milhares de usuários, todos também com suporte a várias atividades de negócio. Compondo a complexidade, a rede de TI deve sempre estar disponível, ser responsiva e proteger os recursos de informações preciosas da empresa. Além disso, a rede de TI deve ser flexível o suficiente para dar suporte às necessidades constantes de alteração dos negócios e para adotar novas tecnologias assim que surgem.

Alguns arquitetos claramente se sobressaem e prosperam nessa complexidade. Tivemos sorte de trabalhar ao lado de grandes analistas e arquitetos durante nossas carreiras. Com reflexão nessas experiências, analisamos o que realmente torna um arquiteto alguém excepcional.

Sem exceção, todos os grandes arquitetos dominam a capacidade de conceituar uma solução em níveis distintos de abstração. Organizando a solução em níveis distintos, os arquitetos podem concentrar-se em um único aspecto da solução enquanto ignoram temporariamente todas as complexidades restantes. Depois de estabilizarem essa parte da solução, podem avançar para outros aspectos, evoluindo continuamente e aperfeiçoando as camadas em um modelo coeso; um que possa finalmente ser implementado.

A maioria dos desenvolvedores de software sabe que deve decompor a solução em níveis de abstração. No entanto, isso é muito difícil de aplicar na prática em projetos reais. Após encontrar a primeira barreira, fica fácil abandonar esses níveis na pressa para iniciar a codificação. Grandes arquitetos superam desafios e são disciplinados para manter os níveis durante todo o ciclo de vida do projeto. Eles sabem que se não o fizerem poderão afogar-se na complexidade.

Este artigo apresenta as técnicas para aplicar níveis de abstração a soluções de TI. Primeiro iremos demonstrar a abordagem por meio de um simples exemplo e então iremos propor uma estrutura de artefatos de sistema baseada em níveis formais de abstração.

Níveis de Abstração: Armas Avançadas para todos os Engenheiros

Outras disciplinas da engenharia, como a engenharia civil, estão competindo com a complexidade aproveitando os níveis de abstração durante séculos. Estudemos como outras disciplinas mais maduras de engenharia aplicam os níveis de abstração, iniciando com a engenharia elétrica, que cria sistemas de computadores em contínuo crescimento com uma complexidade também crescente a cada nova geração.

Engenheiros de Hardware

Designers de sistemas modelam sistemas de computadores usando níveis de abstração. Cada nível é bem-definido e fornece uma perspectiva distinta do sistema. Muitos sistemas são criados com três níveis primários – Sistema, Subsistema e Componente, conforme exibido na Figura 1.

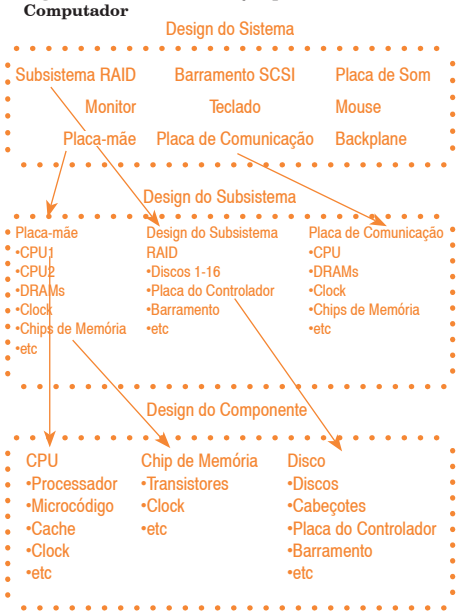
A divisão em camadas permite aos engenheiros integrar uma enorme complexidade em um único sistema computacional. É impossível entender um computador ao analisar simplesmente suas partes. Existem aproximadamente 25.000.000 de transistores em um único chip Intel Itanium®!

Essa abordagem de se dividir a complexidade em camadas de abstração claramente não é exclusiva de disciplinas relacionadas à área de TI. Uma abordagem similar é usada em inúmeras outras disciplinas, desde a engenharia da aeronáutica até a microbiologia.

Princípios Fundamentais ao Aplicar Níveis de Abstração

Todos os engenheiros seguem esse conjunto fundamental de princípios ao aplicar níveis de abstração. Esses princípios também são verdadeiros ao aplicar níveis de abstração a softwares.

Figura 1. Níveis de Abstração para um Sistema de Computador



"Organizando a solução em níveis distintos, os arquitetos podem concentrar-se em um único aspecto da solução enquanto ignoram pelo momento todas as complexidades restantes."

Os três níveis de abstração são definidos da seguinte forma:

Nome do Nível	Escopo do Nível
Sistema	O engenheiro de computador cria o sistema integrando vários subsistemas, como um backplane, placas de circuito, um chassi, dispositivos internos (unidades de CD/DVD e de disquete) e dispositivos externos (monitor, teclado e mouse). O engenheiro pensa no sistema em termos de seus subsistemas, suas interfaces e suas interconexões. Cada interface de subsistema é documentada como uma especificação formal ao designer do subsistema.
Subsistema	O engenheiro do subsistema cria o subsistema integrando componentes. Por exemplo, o designer da placa-mãe projeta a placa-mãe integrando componentes como chips de memória, chips DMA e um chip de CPU. Da mesma forma, o engenheiro do monitor cria o monitor integrando componentes como a placa de vídeo e o CRT. O engenheiro do subsistema pensa no subsistema em termos de seus componentes, suas interfaces e suas interconexões. Cada interface de componente é documentada como uma especificação formal ao engenheiro do componente.
Componente	O engenheiro do componente cria o componente montando e integrando subcomponentes. Por exemplo, o designer do chip de memória projeta o chip como uma rede complexa de circuitos integrados.

Nome do Nível	Escopo do Nível
Domínio	– A empresa é o ator central da "caixa preta". – Modela os negócios pela perspectiva dos atores externos comerciais. – Modela apenas as interações de negócio. Omite os meios de comunicação.
Processos de Negócios	– Modela os fluxos de trabalho de processos de negócio que são a realização das interações de negócio no nível de domínio. – O sistema serve como o ator central da "caixa preta". – Modela os processos de negócio pela perspectiva dos atores externos do sistema. Inclui os meios de comunicação para completar as transações comerciais.
Lógico	– Modela o design interno do sistema. – Os componentes mais importantes do sistema funcionam como os atores principais. – Modela o comportamento do sistema pela perspectiva da parte interna da "caixa preta" do sistema.
Físico	– Modela a estrutura física da implementação.

O número e o escopo dos níveis estão bem-definidos – Para que os engenheiros colaborem em um sistema complexo, todos os membros da equipe devem compartilhar a mesma compreensão dos níveis. Para que os designers tomem decisões de criação, eles devem arquivar essas decisões no nível apropriado de detalhe.

Várias visões dentro de cada nível – A complexidade dentro de um único nível pode proporcionar muita informação de uma só vez. Nesse caso, os engenheiros apresentam o design dentro de um único nível por meio de várias visões. Cada visão apresenta um aspecto particular do design e ainda assim permanece no mesmo nível de abstração. Por exemplo, o engenheiro da placa-mãe cria uma visão para cada camada da placa, modelando o design dos caminhos de conexão para essa camada.

É necessário manter consistência entre os níveis – Para que o sistema funcione conforme o esperado, cada camada subseqüente deve ser o aprimoramento adequado de sua camada pai. Se o designer do sistema de computador comutar de um barramento IDE para um barramento SCSI, então as especificações da interface para todos os dispositivos também deverão comutar para SCSI. Se os níveis não estiverem sincronizados, o sistema não terá o desempenho previsto no nível superior.

Aplique Níveis de Abstração a Soluções de TI

Agora que examinamos como outras disciplinas aplicam níveis de abstração, apliquemos a técnica a soluções de TI[1]. As próximas seções apresentam técnicas para aplicar níveis de abstração para modelar os requisitos, o design e a implementação de uma típica aplicação de TI. As técnicas são apresentadas por meio de um exemplo simples e instrucional de um sistema de pedidos on-line para um revendedor hipotético. Em nosso exemplo, não incluímos apenas a arquitetura, mas expandimos o escopo para incluir os requisitos de sistema e o contexto de negócio conforme definido pela indústria do varejo.

[1] Muitos aplicaram com sucesso níveis de abstração ao software. Ed Yourdon e Tom DeMarco propuseram uma análise estruturada e um design de sistema estruturado em 1979. Várias agências do governo norte-americano padronizaram-se na norma 2167A do Departamento de Defesa, que requer que os sistemas sejam compostos de uma hierarquia de itens de configuração de hardware e software. A comunidade DBA freqüentemente aplica níveis de detalhe para modelar bancos de dados relacionais. Em particular, o conjunto de ferramentas Bachman e a metodologia de engenharia de informação de James Martin (IEM - Information Engineering Methodology) modelam bancos de dados logicamente, e então fisicamente. Uma pesquisa no Google sobre "níveis de abstração de software" oferece vários resultados; no entanto, a maioria pertence à comunidade acadêmica e parece concentrar-se em linguagens formais de computador.

Estrutura Simples: Quatro Níveis de Abstração

Nosso exemplo simples define os quatro seguintes níveis de abstração para uma solução de TI:

Dentro de cada nível, apresentamos tanto a visão dinâmica como a visão estática do comportamento desse nível em particular. Enquanto a visão dinâmica modela as mensagens entre os objetos, a visão estática modela a estrutura e as relações entre os objetos.

Nível de Abstração do Domínio

Aplicando as regras de escopo descritas anteriormente, o revendedor serve como o ator central da caixa preta no nível do domínio. O cliente serve como o ator externo. O nível do domínio é modelado pela perspectiva do cliente. Somente as interações de compra são modeladas. Os meios de comunicação usados para completar a compra não estão incluídos nesse nível, mas são introduzidos no nível do processo de negócio.

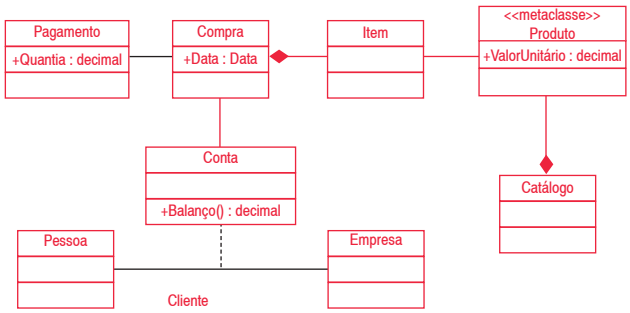
Visão Dinâmica

A visão dinâmica dentro do nível do domínio modela as interações entre o cliente e o revendedor. A figura seguinte resume o contexto do domínio e contém uma narração simples de caso de uso das interações de negócio.

Visão Estática

A visão estática do nível do domínio modela a estrutura da classe e suas relações dos objetos apresentados no caso de uso. Em outras palavras – "Quais objetos o cliente precisa compreender para concluir a transação de compra nesse nível da abstração?" *A Figura 3* apresenta o diagrama de classe para a visão estática do nível do domínio.

Figura 3. Visão estática do nível do domínio para a aquisição de itens de algum revendedor



O cliente é uma instância de *Pessoa*. A relação entre o cliente e o revendedor é considerada uma *Conta*. Todas as *Compras* estão associadas à *Conta* do cliente. A Compra é associada a cada um dos Itens sendo adquiridos. Cada Item pertence a um Produto específico, em que o Produto segue o padrão da metaclasse. Instâncias de *Produto* encontram-se em suas próprias classes de bens. A modelagem de um *Produto* como metaclasse faz com que nosso modelo seja mais flexível, visto que a inclusão de *Produtos* adicionais ao *Catálogo* é um processo puramente orientado por dados e não causa impactos no modelo de classe. Arredondando as classes, cada *Pagamento* é associado a sua *Compra*.

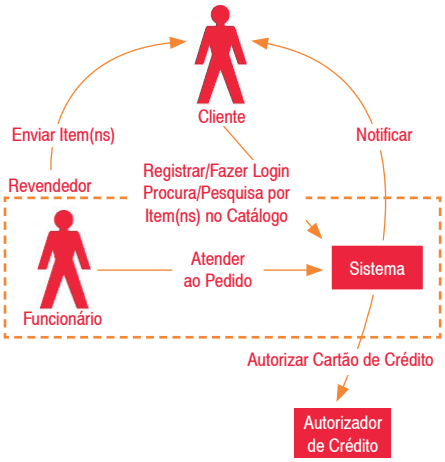
Figura 2. Visão dinâmica do nível do domínio para a aquisição de itens de algum revendedor

Itens de compra
1. O cliente observa o catálogo de itens do revendedor
2. O cliente seleciona um ou mais itens
3. O cliente paga o revendedor pelos itens

Como se pode ver, o modelo nesse nível é representativo para a maioria dos revendedores – on-line ou tradicionais, pequenos ou grandes. Isso ilustra porque o Modelo de Domínio [Industrial] deve realmente definir a empresa como o ator central da caixa preta. Empresas na mesma indústria tendem a oferecer suporte ao mesmo conjunto de interações de negócio com seus atores externos. Além disso, os modelos de domínio excluem os processos de negócio específicos da empresa, visto que podem variar amplamente entre as empresas da mesma indústria.

O nível do domínio concentra-se estritamente nas interações de negócio visualizadas pelas perspectivas do ator externo. Devemos ter cuidado para não incluir mecanismos de implementação para cumprir as interações. Esses detalhes pertencem ao próximo nível de abstração. Portanto, neste caso, iremos modelar somente a procura, a seleção, a compra e o pagamento. Não iremos modelar como essas interações são cumpridas – por telefone, correio, e-mail, aplicação web, pessoalmente, cheque, cartão de crédito ou em dinheiro.

Figura 4. Visão dinâmica do nível do processo de negócio para a aquisição on-line de itens de algum revendedor



Aquisição On-line de Itens

1. O cliente visita o site do revendedor
2. O cliente observa o catálogo on-line do revendedor
3. Alternativamente, o cliente realiza uma busca por palavras-chave no catálogo
4. O cliente seleciona o(s) item(ns)
5. O cliente faz login em sua conta on-line
6. O cliente cria um pedido para o(s) item(ns)
7. O cliente paga pelos itens com cartão de crédito
8. O funcionário verifica no sistema se há pedidos não atendidos
9. O funcionário atende ao pedido retirando os itens do inventário, embalando-os e enviando-os ao endereço de envio contido no pedido
10. O funcionário atualiza o sistema com as informações de envio que são subtraídas do inventário de itens
11. O sistema envia automaticamente uma notificação por email para o cliente indicando o status do envio

Variações

- O cliente pode fazer login em qualquer momento antes de criar o pedido
- Na etapa 5, o cliente pode registrar uma nova conta de usuário caso ainda não possua uma
- Na etapa 7, o autorizador de crédito pode rejeitar o pagamento em cartão de crédito caso a autorização falhe

Processo de Negócio de Abstração
O próximo nível de abstração modela os processos de negócio da empresa para realizar as interações que foram capturadas no nível do domínio. O nível do sistema "dá um zoom" dentro da caixa preta da empresa e identifica todos os funcionários e sistemas que colaboram para executar a transação comercial. Nesse nível, o sistema a ser desenvolvido serve como o ator central da caixa preta.

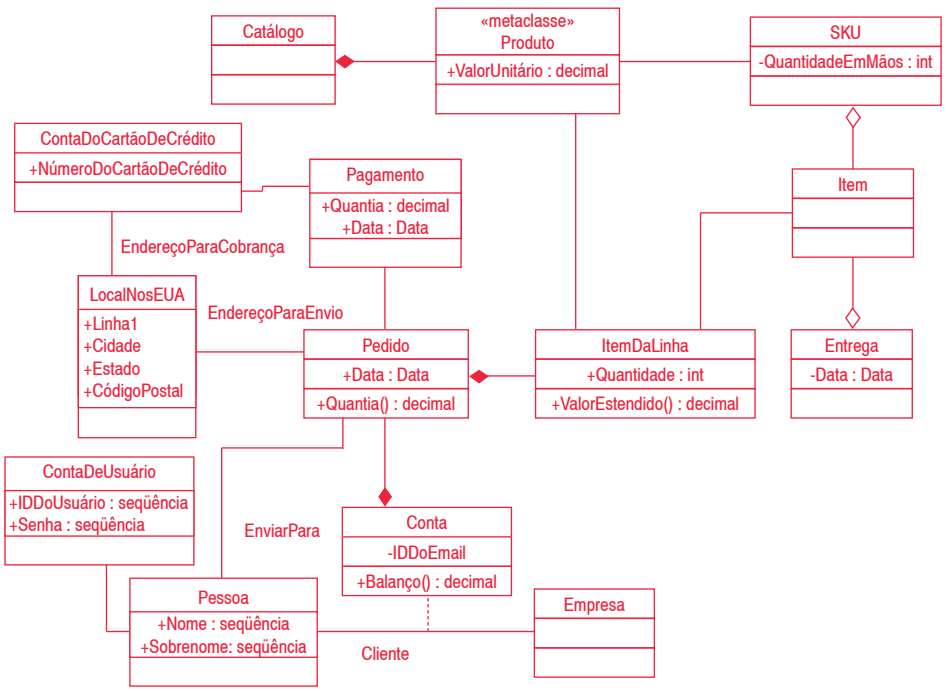
Aplicando as regras de escopo para o nível do sistema, o sistema de pedidos on-line serve como o ator central da caixa preta. O cliente e o funcionário servem como os atores externos. O nível do sistema é modelado pela perspectiva do cliente e do funcionário. O cliente realiza a compra on-line. O pagamento é feito com cartão de crédito. O pedido é atendido com o envio dos itens ao endereço de envio do cliente. Uma notificação desse envio é enviada por email.

Visão Dinâmica
A visão dinâmica reproduz a transação de compra do nível do domínio, neste momento expondo os processos de negócio internos do revendedor. A Figura 4 resume o contexto do processo de negócio e contém uma simples narração de caso de uso das interações entre o sistema e seus atores.

Visão Estática
A visão estática nesse nível aperfeiçoa o modelo da classe para capturar os objetos testemunhados nos casos de uso do nível do processo de negócio. Em outras palavras – "Quais objetos o cliente e o funcionário precisam compreender para criar um pedido on-line e atender ao pedido?"

A Figura 5 apresenta o diagrama de classe para a visão estática do processo de negócio.

Figura 5. Visão estática do nível do processo de negócio para a aquisição on-line de itens de algum revendedor



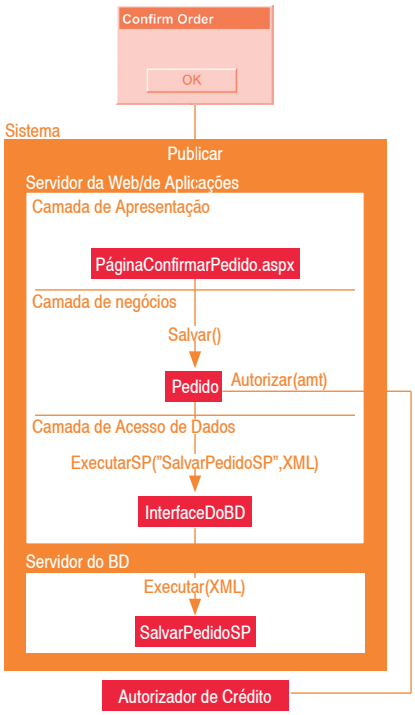
"Para que os engenheiros colaborem com um sistema complexo, todos os membros da equipe devem compartilhar a mesma compreensão dos níveis."

Nós aperfeiçoamos o modelo de classe de domínio para capturar a perspectiva nesse nível da abstração. As abstrações da *Pessoa*, da *Conta* e da *Empresa* continuam as mesmas, bem como o Catálogo e o Produto. No entanto, o evento de *Compra* abstrato do modelo do domínio é substituído por um *Pedido*. *Pedidos* contêm *ItensDaLinha* que estão associados ao *Produto* no *Catálogo*. Como este nível modela o processo de nogócio interno da empresa, é necessário capturar o inventário em mãos (um atributo da unidade de estoque particular *(SKU)* que representa um inventário de itens em um local particular). Também modelamos a *ContaDeUsuário* do cliente, que fornece acesso ao sistema on-line. O *Pagamento* é realizado por meio da *ContaDoCartãoDeCrédito*. O *Local* representa o ponto geográfico dentro de um país e serve como o endereço para cobrança e também como o endereço para entrega do *Pedido*. A *Entrega* contém *Itens* incluídos na *Entrega*.

O nível de abstração do sistema geralmente requer muita criatividade porque é nesse nível que inventamos maneiras de modernizar os processos de negócio. Ao fazer isso, é comum notar uma única transação de nível de domínio usando vários meios diferentes no nível do processo de negócio. Por exemplo, é possível efetuar a compra on-line, pelo telefone, enviando o pedido por correio ou fax, ou pessoalmente na loja de varejo. Cada uma dessas opções precisa ser modelada no nível do processo de negócio. Observe que embora o Autorizador de Crédito seja um ator externo ao revendedor, ele é introduzido nesse nível por ser necessário somente para implementar um processo de negócio que aparece primeiramente nesse nível.

Por último, observe que o sistema é independente de tecnologias. Nosso sistema de compras on-line pode ser implementado com qualquer tecnologia web. A seleção de tecnologias dentro da caixa preta do sistema é uma decisão arquitetônica.

Figura 6. Visão dinâmica do nível lógico para a aquisição on-line de itens de algum revendedor



Confirmar Pedido

1. O cliente revisa o resumo do pedido e confirma enviando o formulário.
2. O ASP.Net publica a PáginaConfirmarPedido que recupera o pedido atual e chama Pedido.Salvar().
3. Pedido.Salvar() valida as regras de negócio do pedido. Se alguma for violada, Salvar envia uma lista de erros.
4. Do contrário, Salvar() chama o autorizador de crédito para autorizar a quantia do pagamento.
5. Se for bem-sucedida, Salvar() serializa todas as partes do Pedido e chama a Camada de Acesso a Dados para executar SalvarPedidoSP.
6. A Camada de Acesso a Dados encaminha a solicitação ao procedimento armazenado.
7. SalvarPedidoSP salva o pedido no banco de dados.

Nível de Abstração Lógico

O nível lógico "dá um zoom" dentro da caixa preta do sistema, expondo o design de alto nível do sistema. O arquiteto seleciona a tecnologia e define a estrutura de alto nível do sistema. Em nosso exemplo simples, o sistema é composto de um servidor IIS/ASP.Net que hospeda a apresentação, as camadas de lógica de negócio e de acesso a dados e um servidor de banco de dados SQL Server que hospeda os dados persistentes.

Visão Dinâmica

A visão dinâmica no nível lógico rastreia o fluxo de mensagens pelos principais componentes do sistema. Como um exemplo, a *Figura 6* rastreia o fluxo ao enviar o formulário web ConfirmarPedido.

Visão Estática

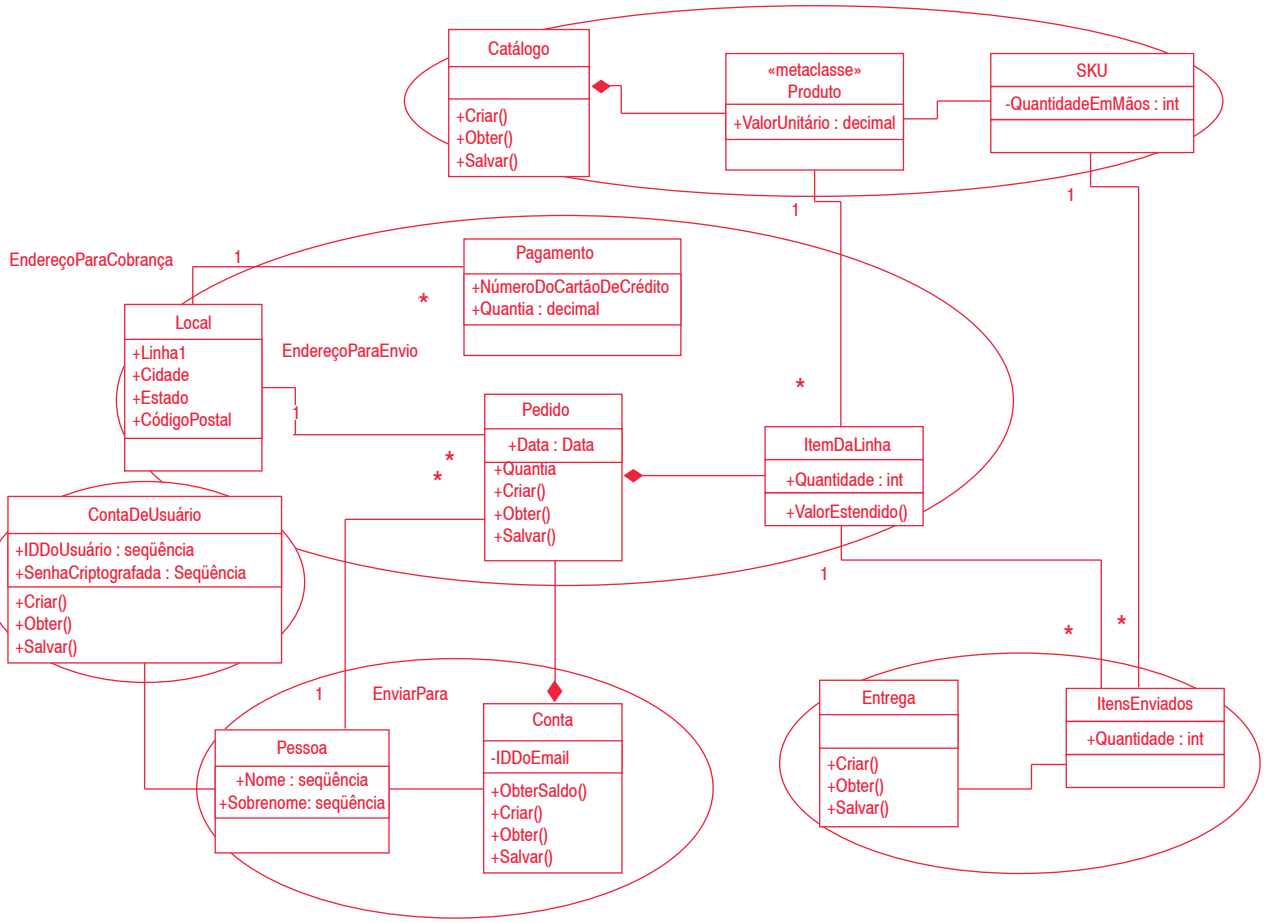
A visão estática neste nível também comuta nossa perspectiva para a parte interna do sistema. Enquanto o nível do processo de negócio modela as abstrações do mundo real que aparecem nos processos de negócio, este nível modela as abstrações conforme devem ser representadas dentro do sistema. No sistema real, o arquiteto deve criar as classes para cada camada de software (apresentação, lógica de negócio e acesso a dados). Para manter este artigo breve, a Figura 7 apresenta apenas o design estático para a camada de negócio de forma a mostrar como as abstrações do nível do sistema foram aperfeiçoadas para o design.

O arquiteto aperfeiçoa as classes do nível do sistema para criar a interface da camada de negócio.

Todas as contas e clientes no sistema pertencem ao revendedor, portanto não é prático criar uma única instância de Empresa e associá-la a

"Os modelos de domínio excluem os processos de negócio específicos da empresa, visto que podem variar amplamente entre as empresas da mesma indústria."

Figura 7. Visão estática do nível lógico para a aquisição on-line de itens de algum revendedor



todas as contas. Por isso, a Empresa é omitida neste nível. Em vez de criar uma instância separada para cada ContaDoCartãoDeCrédito, simplesmente armazenamos o número do cartão de crédito e o endereço para cobrança com o Pagamento. Além disso, não é prático para o sistema criar uma instância para cada Item vendido, assim o Item é removido do modelo e, em vez disso, o modelo rastreia a quantidade de itens pedidos no ItemDaLinha e a quantidade de itens enviados à nova classe ItensEnviados.

O arquiteto também define a granularidade dos serviços que a camada de negócio expõe. Para este exemplo, a camada de negócio exporta os serviços Criar, Ler, Atualizar e Excluir (CRUD) de Conta, ContaDeUsuário, Pedido, Entrega e Catálogo. As elipses indicam a granularidade CRUD.

Observe que embora as classes neste nível não sejam um superconjunto apropriado das classes de processos de negócio, o arquiteto chega até esse design por meio do aperfeiçoamento direto das classes de processos de negócio, alterando a perspectiva de fora para dentro do sistema.

Nível de Abstração Físico

O nível de abstração físico captura a estrutura da implementação do sistema. O sistema é implementado como uma rede de nós, e cada nó é configurado com hardware e software. As três camadas de software na visão lógica (apresentação, lógica de negócio e acesso a dados) estão fisicamente implementadas como código e implantadas nesses nós. As classes persistentes na visão lógica são

armazenadas fisicamente nas tabelas relacionais em um banco de dados do SQL Server.

Visão Dinâmica
A visão dinâmica rastreia o fluxo de mensagens pelos nós da configuração física. O HTTP post ConfirmarPedido flui do navegador do cliente, através da Internet e do firewall do revendedor, ao servidor web onde o Windows encaminha ao IIS, que o passa ao ASP.Net, que então enviará ConfirmarPedido.aspx. Felizmente, as ferramentas modernas de desenvolvimento isolam-nos de maior parte da rede física. Arquitetos, no entanto, precisam compreender a camada física para evitar gargalos na rede e exposições de segurança.

Visão Estática
A visão estática (Figura 7) aperfeiçoa as classes persistentes na visão lógica para sua representação física. Em nosso exemplo de venda, as classes da camada de negócio estão armazenadas nas tabelas do SQL Server.

As classes são mapeadas para as tabelas relacionais e os atributos são implementados como colunas. As relações um-para-um e as relações um-para-muitos são implementadas usando-se uma chave externa. A concorrência otimista é implementada atribuindo-se um campo datahora a cada classe-pai CRUD.

Ao criar o nível lógico, o arquiteto concentra-se principalmente em implementar a funcionalidade do sistema. Confiante de que a funcionalidade do sistema foi coberta, o arquiteto pode concentrar-se em melhorar a implementação no nível físico.

Desenvolva os Níveis pelas Repetições
Depois de ter estabelecido esta estrutura, o arquiteto desenvolve a solução sobre várias repetições. Cada repetição incorpora funcionalidades adicionais – faturas, pedidos pendentes, pedidos feitos pessoalmente, pedidos feitos pelo telefone e outros. Em cada caso, o arquiteto atualiza o nível apropriado de abstração e então refina as atualizações para o nível de implementação física.

Reveja os Princípios Fundamentais dos Níveis de Abstração
Façamos um teste de nosso exemplo com relação aos princípios fundamentais dos níveis de abstração.

O número e o escopo dos níveis estão bem-definidos
Existem quatro níveis distintos – A caixa preta da empresa, a caixa preta do sistema, o design lógico dentro do sistema e a implementação física.

Várias visões dentro de cada nível
Neste exemplo simples, apresentamos uma visão dinâmica e uma estática em cada nível.

É necessário manter consistência entre os níveis
Se alguma alteração for feita no modelo do domínio, o impacto das alterações deve fluir para os níveis inferiores. Por exemplo, se algum revendedor decidir oferecer contratos de manutenção para seus produtos, o analista deverá adicionar ContratoDeManutenção ao modelo do domínio e aperfeiçoá-lo para sua representação física. A sincronização de todos os níveis é fundamental para a manutenção de grandes sistemas. Conforme as solicitações de aperfeiçoamento são enviadas, o analista efetua uma avaliação do impacto no nível apropriado de detalhe. Alguns aperfeiçoamentos causam impacto no nível do domínio (e, portanto, em todos os níveis subseqüentes). Outros causam impacto somente no nível físico.

Figura 8. Visão estática do nível físico para a aquisição on-line de itens de algum revendedor

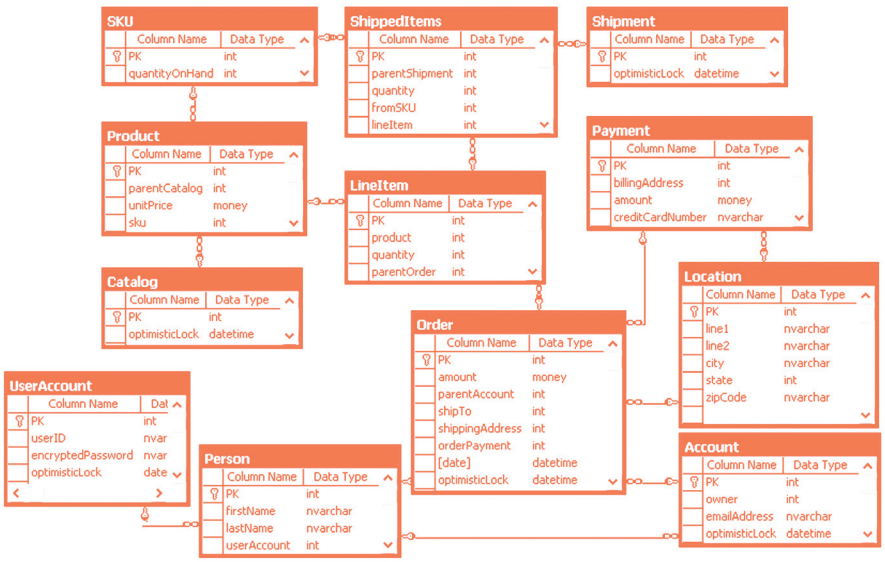


Figura 9. Configuração RUP organizando artefatos do projeto em níveis de abstração bem-definidos

Nome do Nível	Escopo do Nível
Domínio	A empresa é o ator central da "caixa preta". Modela os negócios pela perspectiva do ator externo comercial. Modela apenas as interações de negócio. Não inclui meios de comunicação.
Visão do Projeto	Missão do projeto, objetivos de negócio do projeto, retorno do investimento no projeto.
Processo de Negócio	Modela os fluxos de trabalho de processos de negócio que são a realização das interações de negócio do nível do domínio. O sistema serve como o ator central da "caixa preta". Modela os processos de negócio pela perspectiva dos atores externos do sistema. Inclui os meios de comunicação para completar as transações comerciais.
Especificação da Interface de Usuário	Design e protótipo da interface de usuário da funcionalidade do sistema. Demonstra como usuários do sistema podem realizar fluxos de trabalho de processos de negócio anteriores.
Requisitos Detalhados	Especifica os detalhes do nível mais baixo que representam a interface externa do sistema. Por exemplo, "Os códigos postais dos EUA devem ser mascarados como xxxxx ou xxxxx-xxxx". Especifica os requisitos de proficiência – desempenho, disponibilidade, segurança, internacionalização, configuração, escalabilidade e flexibilidade.
Arquitetura	Dentro da "caixa preta" do sistema.
Implementação	Esquemas de bancos de dados, códigos-fonte, dados de referência, arquivos de configuração.

Visão Lógica	Visão da Concorrência	Visão da Segurança	Visão da Implantação	Visão dos Componentes	Vis'ao dos Dados
Projeto Lógico	Design da Concorrência	Design da Segurança	Configuração da Rede	Interface do Componente	Modelo de Dado Lógico
	Implementação da Concorrência	Implementação de Segurança	Configura;'ao da Caixa	Implementação de Componentes	Modelos de Dados Físicos

Escalando os Níveis para Oferecer Suporte a Soluções Empresariais
Agora que apresentamos um exemplo simples com os quatro níveis de abstração, façamos o ajuste de escala da abordagem para dar suporte a soluções para empresas de TI. A Figura 9 apresenta a configuração RUP (Rational Unified Process) que organiza artefatos de projeto em níveis bem-definidos de abstração.

Os níveis da tabela encontram-se descritos a seguir.

Domínio – O nível do domínio captura o contexto de negócio para um determinado projeto.

Visão do Projeto – A visão do projeto comunica o impacto que o sistema trará aos negócios. Isso quantifica esse impacto em uma análise de retorno do investimento. A visão do projeto representa o nível mais alto de abstração do projeto.

Processo de Negócio – O nível do sistema modela os processos de negócio dentro da empresa. No caso de organizações extremamente complexas, esse nível pode ser dividido em subníveis – divisão, interdepartamental e intradepartamental.

Especificação da Interface de Usuário – A especificação da interface de usuário cria a interface que realiza o processo de negócio. Esse processo é composto de um documento de design e um protótipo funcional da interface do usuário.

Requisitos Detalhados – Os requisitos detalhados especificam o mais baixo nível de abstração dos requisitos do sistema. Também incluem detalhes como os formatos de tipos de dados e as regras de negócio detalhadas. Além disso, também contêm os requisitos de proficiência, como desempenho, disponibilidade, segurança, internacionalização, configuração, escalabilidade e flexibilidade.

"Não mais sobrecarregamos os usuários de negócio com uma única especificação funcional, monolítica."

Arquitetura – A arquitetura do sistema é organizada em seis visões –
– *Lógica:* Define as camadas de software e as principais abstrações que executam a funcionalidade do sistema.
– *Concorrência:* Captura os aspectos paralelos do sistema, incluindo transações, farms de servidores e contenção de recursos.
– *Segurança:* Define a abordagem para autenticação, autorização, proteção e segredos e *journaling*.
– *Implantação:* Define a topologia da rede e a configuração de implantação do sistema.
– *Componente:* Define os componentes do sistema, suas interfaces e dependências.
– *Dados:* Define a estrutura do design dos dados persistentes.

Benefícios
A organização dos artefatos do sistema em níveis distintos de abstração proporciona vários benefícios:
– Separa os requisitos do sistema em três níveis distintos de abstração – Processos de Negócio, Especificação da Interface de Usuário e Requisitos Detalhados. Não mais sobrecarregamos os usuários de negócio com uma única especificação funcional, monolítica. Em vez disso, comunicamos os requisitos do sistema em três níveis aperfeiçoados de detalhe.
– Analistas e arquitetos são capazes de gerir a complexidade em um único modelo integrado do sistema.
– Os arquitetos podem concentrar-se em um único aspecto do sistema e integrar essas decisões na solução geral.
– Os níveis de abstração formam a estrutura dos artefatos do sistema. Por exemplo, o documento de arquitetura de software dedica uma subseção para cada visão.
– Os níveis de abstração fornecem uma forma de acompanhar a evolução dos requisitos para o design e para a implementação. Essa capacidade permite que uma equipe realize uma avaliação de impacto precisa ao testar as solicitações de alteração.
– Depois de desenvolver vários sistemas usando a mesma estrutura, padrões surgem em cada nível de abstração. As organizações podem catalogar esses padrões e outras práticas recomendadas dentro de cada nível de abstração. Esse catálogo de práticas recomendadas serve como a base do programa de melhoramento de processos.

Resumo

Todas as disciplinas da engenharia aplicam níveis formais de abstração para lidar com a complexidade. Isso não é exceção no caso do software. Para realizar os benefícios dos níveis de abstração, os projetos devem:

– Identificar formalmente as camadas, cada uma com um escopo bem-definido;
– Dividir a complexidade dentro de um nível em várias visões;
– Manter a consistência entre os níveis.

Este artigo demonstrou como aplicar níveis de abstração por meio de um exemplo simples, e então escalou a abordagem para dar suporte a soluções de empresas de TI. Ofereceu também uma estrutura de configuração RUP que organiza os artefatos do sistema em níveis bem-definidos de abstração.

Auto-avaliação

O seu projeto atual aplica níveis de abstração?
Esses níveis estão bem-definidos?
A equipe do projeto compreende bem esses níveis?
Se a complexidade torna-se muito grande dentro de um nível, a equipe costuma dividi-lo em visões?
A equipe mantém consistência entre os níveis?
O seu projeto pode beneficiar-se dos níveis de abstração?

Os grandes arquitetos seguem esses princípios de forma instintiva. O restante de nós deve aplicar conscientemente os níveis de abstração e exercitar a disciplina para manter os níveis durante todo o ciclo de vida do projeto.

Fontes

Cockburn, Alistair. *Writing Effective Use Cases*. New Jersey: Addison-Wesley, 2001

Kroll, Per e Kruchten, Philippe. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Boston MA: Pearson Education e Addison- Wesley, 2003

DeMarco, Tom e Plauger, P J. *Structured Analysis and System Specification*. Prentice Hall PTR, 1979

É possível obter uma cópia on-line da norma 2167A do Departamento de Defesa no site http://www2.umassd.edu/SWPI/DOD/MIL-STD-2167A/DOD2167A.html.

Sobre os autores

Don Awalt
Presidente e fundador da RDA Corporation AWALT@rdacorp.com

Don Awalt é o presidente e o fundador da RDA Corporation, fundada em 1988 como uma firma de engenharia de software personalizada com escritórios em Washington DC, Baltimore, Atlanta, Filadélfia e Chicago. A empresa, uma Parceira com Certificado Gold da Microsoft, está concentrada no desenvolvimento de sistemas web e rich client que usam o .NET Framework. Don é atualmente o diretor regional da Microsoft em Washington DC e foi anteriormente o diretor regional fundador na Filadélfia. Don é um participante freqüente de eventos da indústria, incluindo eventos da Tech Ed, Developer Days, MSDN e também de vários eventos sobre o SQL Server e o Windows. Também foi um editor contribuinte para a SQL Server Magazine e para a PC Tech Journal Magazine, além de ter escrito várias outras publicações. As áreas de especialidade específicas de Don incluem Serviços Web, SQL Server, a evolução das linguagens de programação modernas e muito do trabalho de processo e arquitetura visto no PAG (Grupo de Arquitetura Prescritiva) da Microsoft.

Rick McUmber
Diretor de Qualidade e de Práticas Recomendadas na RDA McUmber@rdacorp.com

Rick McUmber é Diretor de Qualidade e de Práticas Recomendadas na RDA. Durante 11 anos, ele trabalhou para a IBM e para a Rational Software Corporation respectivamente, desenvolvendo sistemas para o Departamento de Transportes, o Departamento de Defesa, a NASA e o Departamento de Defesa Nacional do Canadá. Desde 1994 trabalha com a RDA no desenvolvimento de soluções de negócio para seus clientes.

Fábricas de Software

Por Jack Greenfield, Microsoft Corporation

Este artigo apresenta brevemente a motivação para Fábricas de Software, uma metodologia desenvolvida na Microsoft. O artigo descreve as forças que estão fazendo a transição de manufatura para fabricação. De forma resumida, as Fábricas de Software são ambientes de desenvolvimento configurados para oferecer suporte ao desenvolvimento rápido de tipos específicos de aplicações. As Fábricas de Software são realmente apenas a etapa lógica seguinte na evolução contínua dos métodos e práticas de desenvolvimento de software. No entanto, prometem alterar a natureza da indústria de software com a introdução de padrões de industrialização.

Melhorando o Desenvolvimento de Software

O desenvolvimento de software praticado atualmente é lento, caro e propenso a erros, freqüentemente gerando produtos com vários defeitos, causando sérios problemas de usabilidade, confiabilidade, desempenho, segurança e outros problemas de qualidade de serviço.

Segundo o Standish Group [Sta94], o comércio nos Estados Unidos gasta por volta de U$250 bilhões em desenvolvimento de software, em aproximadamente 200 projetos a cada ano. Somente 16% desses projetos são concluídos a tempo e dentro do orçamento. Outros 31% são cancelados, principalmente devido a problemas de qualidade, com perdas de cerca de U$81 bilhões. Outros 53% estouram o orçamento em uma média de 189%, com perdas de cerca de U$59 bilhões. Os projetos próximos da conclusão oferecem uma média de apenas 42% dos recursos originalmente planejados.

Esses números confirmam de forma objetiva o que já conhecemos por experiência: que o desenvolvimento de software é um trabalho intenso, consumindo mais capital humano por dólar de valor produzido do que esperamos da indústria moderna.

De forma clara, apesar dessas deficiências, os produtos de desenvolvimento de software obviamente fornecem valor significativo aos consumidores, conforme demonstrado pela tendência, em longo prazo, de um aumento de demanda. Isso não indica que os consumidores estejam completamente satisfeitos, seja com o software que fornecemos ou com a forma como o fornecemos. Meramente indica que eles valorizam o software, tanto que estão dispostos a correr grandes riscos e perdas para tirar proveito dos benefícios fornecidos. Embora essa situação obviamente não seja a melhor, conforme demonstrado pela crescente popularidade da terceirização, ela parece não estar forçando qualquer alteração significativa nos métodos e práticas de desenvolvimento de software do mercado.

Apenas ganhos modestos na produtividade foram gerados durante a última década, talvez os mais importantes deles sejam as linguagens e os padrões codificados por byte e os métodos ágeis. Apesar desses avanços, ainda desenvolvemos software da mesma forma que fazíamos há dez anos. Nossos métodos e práticas ainda não mudaram muito e nem os custos e riscos associados.

No entanto, essa situação está para mudar. A demanda total global de software está projetada para aumentar em uma ordem de magnitude na próxima década, orientada pelas novas forças na economia global como o desenvolvimento da China e o papel crescente do software na infra-estrutura social; por meio de novas aplicações como integração de negócio e computação médica; e por novas tecnologias de plataformas como serviços web, dispositivos móveis e aparelhos inteligentes.

Sem o crescimento comparável na capacidade, parece inevitável que a capacidade total de desenvolvimento de software está destinada a ficar muito abaixo da demanda total até o final da década. É claro que, se as forças do mercado possuem liberdade, isso realmente não acontecerá, visto que o interesse dos fornecedores de software fornecerá a capacidade necessária para atender à demanda.

Enfrentando as Alterações, Novamente

O que irá mudar, então, para fornecer a capacidade adicional? Não é necessária muita análise para ver que os métodos e as práticas de desenvolvimento de software terão de mudar drasticamente.

Visto que a capacidade da indústria depende do tamanho da competência do desenvolvedor e da produtividade de seus membros, a capacidade crescente da indústria exige mais desenvolvedores que utilizam os métodos e as práticas atuais ou um número comparável de desenvolvedores que use métodos e práticas diferentes.

Enquanto a cultura de aprendizagem cultivada durante os últimos dez anos parece ter aumentado com sucesso o número de desenvolvedores competentes e a média de competência do desenvolvedor, a aprendizagem parece não equipar o mercado para satisfazer o nível esperado de demanda por dois motivos:

- Sabemos por experiência própria que nunca existirão mais que alguns programadores completos. Os melhores desenvolvedores são mil vezes mais produtivos do que os piores, mas o pior excede o melhor por uma margem semelhante [Boe81].
- Conforme observado por Brooks [Bro95], a adição de pessoas a um projeto acaba gerando um retorno marginal decrescente. A quantidade de capacidade obtida pelo recrutamento e treinamento de desenvolvedores diminuirá assintoticamente.

"As Fábricas de Software são ambientes de desenvolvimento configurados para oferecer suporte ao desenvolvimento rápido de tipos específicos de aplicações, e prometem alterar a natureza da indústria de software com a introdução de padrões de industrialização."

A solução, portanto, deve envolver alterações em nossos métodos e práticas. Devemos encontrar maneiras para fazer com que os desenvolvedores sejam mais produtivos.

Curvas na Inovação e Mudanças em Paradigmas

Como mercado, já estivemos aqui antes. A história do desenvolvimento de software é um ataque à complexidade e à mudança, com ganhos contrabalançados por perdas, conforme o progresso cria a demanda crescente. Embora grandes progressos tenham ocorrido em meros cinqüenta anos, eles ainda não são constantes. Pelo contrário, seguiram um padrão conhecido de curvas na inovação, conforme ilustrado na *Figura 1* [Chr97].

Figura 1. Curvas na Inovação

Normalmente, uma inovação descontínua estabelece a base para uma nova geração de tecnologias. O progresso na nova base inicialmente é rápido, mas diminui gradualmente, conforme a base se estabiliza e amadurece. No final, a base perde sua capacidade de manter a inovação e chega-se em um plano. Nesse ponto, outra inovação descontínua estabelece outra base para outra geração de novas tecnologias, e o padrão repete-se. Kuhn denomina esse fato como paradigmas fundamentais, e as transições entre eles são mudanças nos paradigmas [Kuh70]. As mudanças nos paradigmas ocorrem nos momentos em que são necessárias alterações existentes para manter o impulso para frente. Agora nós nos encontramos em tal momento.

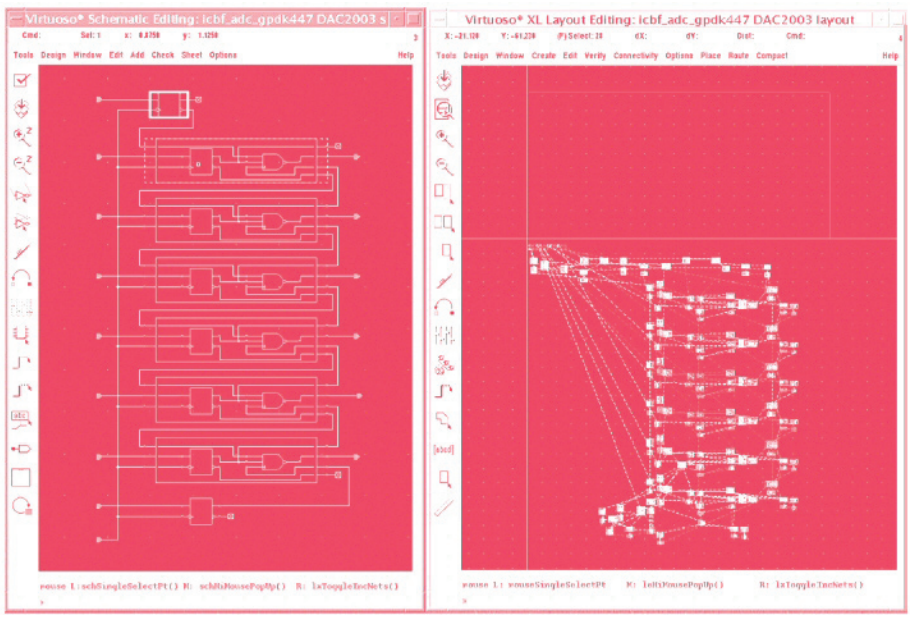
Elevando o Nível de Abstração

Historicamente, as mudanças em paradigmas elevaram o nível de abstração para os desenvolvedores, fornecendo conceitos mais avançados para capturar e reutilizar o conhecimento em plataformas e linguagens. No caso da plataforma, por exemplo, estamos progredindo desde o processamento em batch, pelo terminal burro, pela arquitetura cliente/servidor, pela computação pessoal, por sistemas multicamadas e pela integração de aplicações empresariais, até os serviços flexíveis e assíncronos. No caso da linguagem, progredimos desde a codificação numérica, pelo assembly, pelas linguagens estruturadas orientadas a objetos, até as linguagens e padrões codificados por byte, que podem ser vistos como abstrações baseadas em linguagens. Smith e Stotts resumem essa progressão de forma eloqüente [SS02]:

A história da programação é um exercício na abstração hierárquica. Em cada geração, os designers de linguagem produzem construções baseadas em lições aprendidas na geração anterior e então os arquitetos a utilizam para criar abstrações ainda mais complexas e avançadas.

Eles também apontam que as novas abstrações tendem a aparecer primeiramente em plataformas e então migrar para as linguagens. Estamos agora em um momento nessa progressão no qual a linguagem baseada em abstrações encontra-se bem atrás das abstrações baseadas em plataformas há bastante tempo. Ou, para colocar de forma diferente, estamos agora em um momento no qual

Figura 2. Ferramentas de Design Baseadas em ASIC1



[1] Esta ilustração que apresenta o Virtuoso® Chip Editor e o Virtuoso® XL Layout Editor foi reproduzida com a permissão da Cadence Design Systems, Inc © 2003 Cadence Design Systems, Inc. Todos os direitos reservados. Cadence e Virtuoso são marcas registradas de Cadence Design Systems, Inc.

as ferramentas encontram-se atrás das plataformas há bastante tempo. Com a última geração de tecnologia de plataformas, por exemplo, agora podemos automatizar processos que abrangem vários negócios em qualquer local do planeta por meio de serviços compostos por orquestração, mas ainda "costuramos à mão" cada uma dessas aplicações, como se fossem as primeiras. Criamos grandes conceitos abstratos, como reivindicações de seguros e negócios com securities a partir de pequenos conceitos concretos como loops, strings e inteiros. Organizamos com cuidado e esforço milhões de pequenas partes de código-fonte e recursos inter-relacionados para formar estruturas massivamente complexas. Se o mercado de semicondutores utilizasse uma abordagem similar, seria possível criar processadores massivamente complexos que iriam potencializar essas aplicações por meio de transistores soldados à mão. Em vez disso, são montados componentes pré-definidos chamados ASICs (Circuitos Integrados Específicos de Aplicações) com as ferramentas mostradas na *Figura 2*, e então geradas essas implementações.

Não podemos automatizar o desenvolvimento de software de uma maneira semelhante? Claro que podemos; na verdade, já o fizemos. Os sistemas de gerenciamento de bancos de dados, por exemplo, automatizam o acesso a dados usando o SQL, que fornece benefícios como integração e independência de dados que facilitam a criação e a manutenção de aplicações orientadas por dados. De forma semelhante, grupos de widgets e editores WYSIWYG facilitam a criação e a manutenção de interfaces gráficas de usuário, fornecendo benefícios como independência de dispositivos e montagem visual. Se observarmos de perto como isso foi feito, podemos ver um padrão recorrente.

– Depois de desenvolver vários sistemas em um determinado domínio de problema, identificamos um conjunto de abstrações reutilizáveis para esse domínio, e então documentamos um conjunto de padrões para usar essas abstrações.
– Em seguida, desenvolvemos um runtime, como um framework ou um servidor, para codificar as abstrações e os padrões. Isso nos permite criar sistemas no domínio por meio de instância, adaptação, configuração e montagem de componentes definidos pelo runtime.
– Definimos, então, uma linguagem e criamos ferramentas que oferecem suporte a essa linguagem, como editores, compiladores e depuradores, para automatizar o processo de montagem. Isso nos ajuda a responder com mais rapidez aos requisitos de mudanças, visto que parte da implementação foi gerada e pode ser alterada facilmente.

Esse é o conhecido padrão Language Framework, descrito por Roberts e Johnson [RJ96]. Um framework pode reduzir o custo de desenvolvimento de aplicações em uma ordem de magnitude, mas usar uma pode ser complicado. Um framework define um produto arquetípico, como uma aplicação ou um subsistema, que pode ser completado ou especializado de várias formas para atender às variações nos requisitos. O mapeamento dos requisitos de cada variante de produto no framework é um problema complexo que geralmente requer o conhecimento de um arquiteto ou de um desenvolvedor experiente. Ferramentas baseadas em linguagem podem automatizar esta etapa capturando variações nos requisitos por meio de expressões de linguagem e pela geração do código para o framework.

Industrializando o Desenvolvimento de Software

Outras indústrias aumentaram suas capacidades mudando da manufatura, em que produtos completos são criados desde o início por indivíduos ou pequenas equipes, para a fabricação, em que uma ampla gama de variantes de produto é rapidamente montada a partir de componentes reutilizáveis criados por vários fornecedores e em que máquinas automatizam tarefas simples ou de rotina. Elas padronizaram processos, designs e embalagens utilizando linhas de produtos para facilitar o reuso sistemático e cadeias de suprimentos para distribuir o custo e o risco. Algumas agora são capazes de personalização em massa, em que as variantes do produto são produzidas de forma rápida e com economia sob demanda para satisfazer aos requisitos específicos de clientes individuais.

O Software Pode Ser Industrializado?

Analogias entre software e bens físicos vêm sido discutidas calorosamente. Esses padrões de industrialização podem ser aplicados à indústria do software? Não somos nós especiais ou diferentes das outras indústrias por causa da natureza de nossos produtos? Peter Wegner sintetiza as similaridades e as contradições da seguinte forma [Weg78]:

Produtos de software são, de alguma forma, como produtos tangíveis das disciplinas de engenharia convencionais como pontes, edifícios e computadores. No entanto, existem também determinadas diferenças importantes que oferecem ao desenvolvimento do software algo exclusivo. Como o software é lógico e não físico, seus custos estão concentrados no desenvolvimento e não na produção, e como o software não se desgasta, sua confiabilidade depende de qualidades lógicas como precisão e robustez, em vez das qualidades físicas, como resistência e maleabilidade.

Parte da discussão envolveu uma comparação estilo "maçãs e laranjas" entre a produção de bens físicos, por um lado, e o desenvolvimento de software, por outro. A chave para eliminar a confusão é compreender as diferenças entre a produção e o desenvolvimento, e entre das economias de escala e de escopo.

"Se o mercado de semicondutores utilizasse uma abordagem similar, seria possível criar processadores massivamente complexos que iriam potencializar essas aplicações por meio de transistores soldados à mão."

Para fornecer um retorno de investimento, os componentes reutilizáveis devem ser reutilizados o suficiente para fazer mais do que recuperar seu desenvolvimento seja diretamente, pela redução de custos, ou indiretamente, pela redução de risco ou do tempo de colocação no mercado ou por melhorias na qualidade. Os componentes reutilizáveis são os bens financeiros de uma perspectiva de investimento. Como o custo de tornar um componente reutilizável é geralmente muito alto, níveis lucrativos de reuso são praticamente impossíveis de serem alcançados por acaso. Então é necessária uma abordagem sistemática para o reuso. Isso geralmente envolve a identificação de um domínio em que vários sistemas serão desenvolvidos, a identificação de problemas recorrentes nesse domínio, o desenvolvimento de um conjunto de bens de produção integrados que solucionam esses problemas e então a aplicação deles conforme sistemas são desenvolvidos nesse domínio.

Figura 3. Economias da Escala

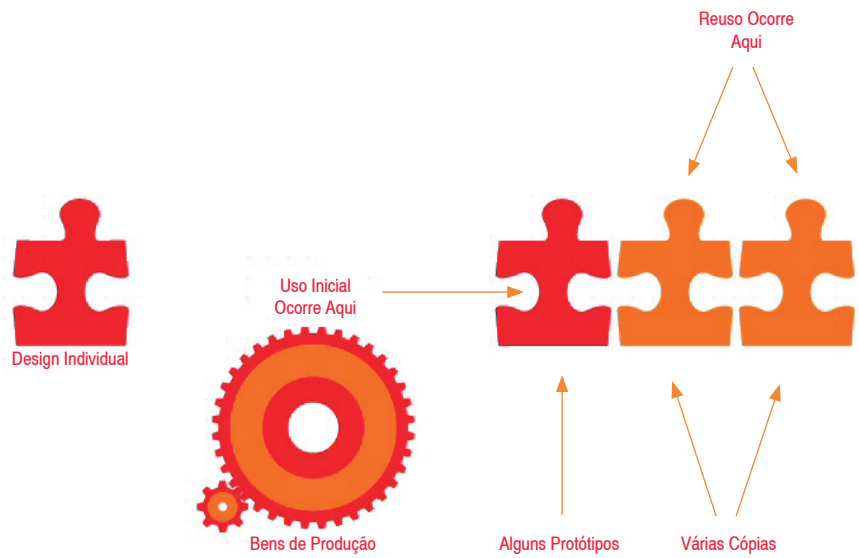
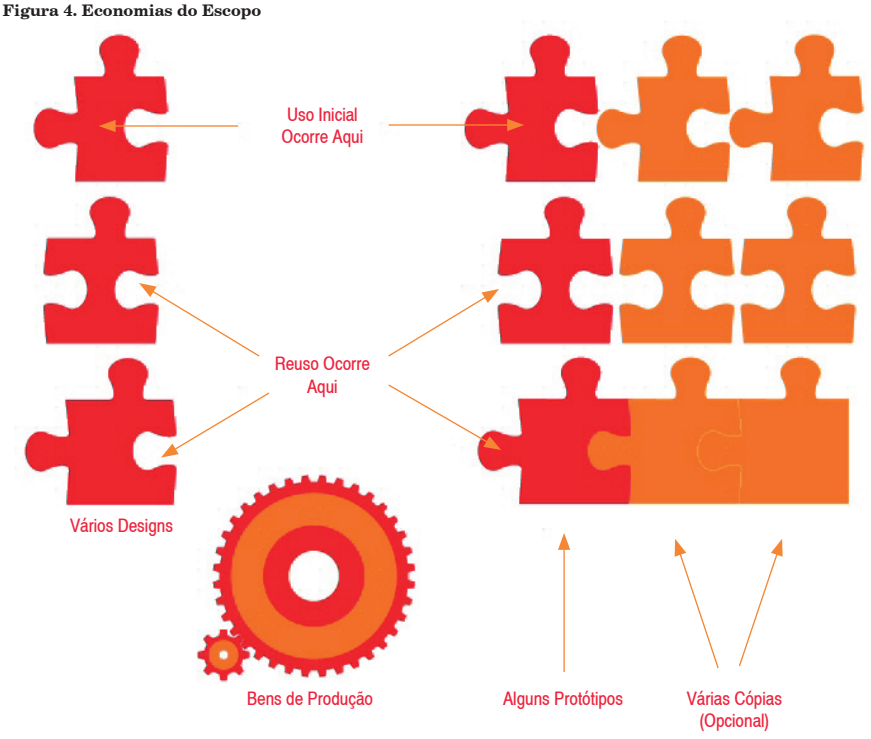


Figura 4. Economias do Escopo



Economias de Escala e de Escopo

O reuso sistemático pode gerar economias tanto na escala quanto no escopo. Esses dois efeitos são conhecidos em outras indústrias. Embora ambos reduzam o tempo e o custo e melhorem a qualidade do produto por meio da produção de vários produtos coletivamente em vez de individualmente, eles diferem-se na forma como produzem esses benefícios.

Economias de escala surgem quando várias instâncias idênticas de um único design são produzidas coletivamente, em vez de individualmente, conforme ilustrado na Figura 3. Elas surgem na produção de itens como parafusos de máquinas, quando os bens de produção, como ferramentas de máquinas, são utilizados para produzir várias instâncias idênticas de produtos. Um design é criado, juntamente com as instâncias iniciais, chamadas protótipos, por um processo com recursos em excesso,

"Outras indústrias aumentaram suas capacidades mudando da manufatura... para a fabricação... em que máquinas automatizam tarefas simples ou de rotina. [E] a chave para encontrar a demanda global é parar de perder o tempo de desenvolvedores capacitados em tarefas simples ou de rotina."

chamado desenvolvimento, realizado por engenheiros. Várias instâncias adicionais, chamadas cópias, são então produzidas por outro processo, chamado produção, realizado por equipamentos e/ou trabalho de baixo custo, para atender à demanda do mercado.

Economias de escopo surgem quando vários designs e protótipos similares, porém distintos, são produzidos coletivamente, em vez de individualmente, conforme ilustrado na *Figura 4*. Na fabricação automotiva, por exemplo, vários designs automotivos similares, porém distintos, são freqüentemente desenvolvidos por meio da composição de designs existentes para subcomponentes, como o chassi, a carcaça, o interior e o sistema de engrenagens, e variações ou modelos são freqüentemente criados por meio de uma variedade de recursos, como motor e kit de acessórios, em designs existentes. Em outras palavras, os mesmos processos, práticas, ferramentas e materiais são utilizados para criar e fazer o protótipo de vários produtos semelhantes, porém distintos. O mesmo é verdadeiro no caso da construção civil, em que várias pontes e arranha-céus compartilham um design comum. No entanto, uma mudança interessante na construção civil é que geralmente uma ou duas instâncias são produzidas de cada design bem-sucedido, portanto, as economias de escala, caso ocorram, são raramente realizadas. Na fabricação automotiva, em que várias instâncias idênticas são geralmente produzidas a partir de designs bem-sucedidos, economias de escopo são complementadas pelas economias de escala, conforme ilustrado pelas cópias de cada protótipo mostrado na *Figura 4*.

É claro que existem diferenças importantes entre software e a fabricação automotiva ou a construção civil, mas elas são semelhantes em determinados momentos.

- Em mercados como desktop de consumidor, em que cópias de produtos, como sistemas operacionais e aplicações de produtividade, são produzidas em massa, o software exibe economias de escala, como na fabricação automotiva.
- Em mercados empresariais, em que as aplicações de negócio desenvolvidas para vantagem competitiva, se forem, raramente são produzidas em massa, o software exibe somente economias de escopo, como na construção civil.

Agora podemos ver onde as maçãs foram comparadas com as laranjas. A produção nas indústrias físicas foi comparada de forma ingênua com o desenvolvimento no software. Não faz sentido procurar por economias de escala em desenvolvimento de qualquer tipo, seja de software ou de bens físicos. Podemos, entretanto, esperar que a industrialização do desenvolvimento de software explore as economias do escopo.

Com o que a Industrialização Irá Parecer?

Presumindo que a industrialização ocorra no mercado do software, com o que ela irá parecer? Não podemos saber com certeza até que ocorra, é claro. Podemos, porém, fazer suposições educadas com base na forma como a indústria de software evoluiu, e em com que a industrialização parece em outros mercados. Claramente, o desenvolvimento de software nunca será reduzido a um mecanismo puramente mecânico e monótono. Pelo contrário, a chave para encontrar a demanda global é parar de perder o tempo de desenvolvedores capacitados em tarefas simples ou de rotina. Devemos encontrar maneiras de fazer melhor uso de recursos preciosos em vez de gastá-los na construção manual de produtos finais que exigirão manutenção ou até mesmo substituição dentro de poucos meses ou anos, quando o lançamento da próxima grande plataforma surgir ou quando as condições do mercado em alteração fizerem com que os requisitos de negócio mudem, o que vier primeiro.

Uma forma de fazer isso é fornecer aos desenvolvedores formas de encapsular seu conhecimento como bens reutilizáveis que outros possam aplicar. Isso é improvável? Os padrões já demonstram reuso de conhecimento limitado, porém eficaz. A próxima etapa é passar da documentação para a automação, usando linguagens, estruturas e ferramentas para automatizar a aplicações de padrões.

O desenvolvimento de semicondutores oferece uma prévia do que o desenvolvimento de software irá lembrar quando a industrialização ocorrer. Isso para não dizer que os componentes de software serão de fácil montagem como os ASICs a qualquer momento. Os ASICs são os produtos altamente evoluídos de duas décadas de inovação e padronização na tecnologia de pacotes e interface. Por outro lado, isso pode levar menos de 20 anos. Temos a vantagem de lidar somente com bits, enquanto o mercado de semicondutores tem o trabalho adicional de projetar materiais físicos usados para a implementação de componentes. Ao mesmo tempo, a natureza efêmera dos bits cria desafios como a proteção dos direitos de propriedade digital, conforme visto na indústria de filmes e música.

Conclusão
Este artigo descreve a incapacidade da indústria de software de atender às demandas projetadas usando os métodos e as práticas atuais. Vários problemas foram brevemente discutidos aqui, sem dúvidas deixando o leitor querendo evidências ou uma discussão mais detalhada. Uma discussão muito mais detalhada é fornecida no livro "Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools", por Jack Greenfield e Keith Short, da John Wiley and Sons.
Também é possível obter mais informações nos sites

http://msdn.microsoft.com/architecture/overview/softwarefactories e

http://www.softwarefactories.com/,

incluindo artigos que descrevem problemas crônicos que impedem a transição da manufatura para a fabricação, as inovações críticas que irão ajudar a indústria a superar esses problemas e a metodologia das Fábricas de Software, que integra inovações críticas.

Referências
1. [Boe81] B Boehm. *Software Engineering Economics*. Prentice Hall PTR, 1981

2. [Bro95] F Brooks. *The Mythical Man-Month*. Addison-Wesley, 1995

3. [Chr97] C Christensen. *The Innovator's Dilemma*, Harvard Business School Press, 1997

4. [Kuh70] T Kuhn. *The Structure Of Scientific Revolutions*. The University Of Chicago Press, 1970

5. [RJ96] D Roberts e R. Johnson. *Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks*. *Proceedings of Pattern Languages of Programs*, Allerton Park, Illinois, setembro de 1996

6. [SS02] J. Smith e D Stotts. *Elemental Design Patterns – A Link Between Architecture and Object Semantics*. Proceedings of OOPSLA 2002

7. [Sta94] The Standish Group. *The Chaos Report*. http://www.standishgroup.com/sample_research/PDFpages/chaos1994.pdf

8. [Weg78] P Wegner. Research *Directions In Software Technology*. Proceedings Of The 3rd International Conference On Software Engineering. 1978

Jack Greenfield
Arquiteto, Microsoft Corporation
jackgr@microsoft.com

Jack Greenfield é um arquiteto de ferramentas e estruturas empresariais na Microsoft. Anteriormente era arquiteto chefe participante do grupo de desktop na Rational Software Corporation, e também fundador e diretor de tecnologia da InLine Software Corporation. Na NeXT, desenvolveu a estrutura de objetos empresariais, agora denominada Apple Web Objects. Palestrante e escritor de renome, também contribuiu para UML, J2EE e as especificações relacionadas a OMG e JSP. É formado em Física pela George Mason University.

Gerenciamento de Identidade e Acesso

Por Frederick Chong, Microsoft Corporation

Resumo
Até o momento, vários responsáveis por decisões técnicas em grandes ambientes de TI ouviram falar sobre os princípios e os benefícios de SOA (Service Oriented Architecture). Apesar disso, pouquíssimas organizações de TI possuem a capacidade de converter a base teórica de SOA em ações de TI práticas.

último ano, poucos arquitetos de soluções de minha equipe tentaram destilar a essência prática de SOA nas seguintes áreas: Gerenciamento de Identidade e Acesso, Gerenciamento de Serviços, Agregação de Entidades e Integração de Processos. Essas quatro principais áreas apresentam desafios técnicos significativos para superar e ainda fornecer as bases críticas de TI para ajudar os negócios a obter os benefícios de SOA.

Note que são nossas interações freqüentes com os arquitetos de empresas que nos permitem conferir, sintetizar e categorizar os desafios práticos de SOA nessas áreas. Nossa equipe organiza os Fóruns Estratégicos para Arquitetos que acontecem várias vezes ao ano, em todo o mundo. Nesses eventos, conduzimos pequenos grupos de discussão para descobrir os pontos problemáticos e a orientação técnica que os clientes estão procurando. Os feedbacks de nossos clientes foram muito consistentes; problemas em gerenciar identidades, agregar dados, gerenciar serviços e integrar processos de negócio foram citados inúmeras vezes como os principais obstáculos para criar organizações mais ágeis e eficientes. Além disso, nossa equipe também conduz projetos de verificação de conceito com clientes para aprofundar-se nos requisitos reais e nos problemas de implementação. É por meio dessa combinação de compromissos amplos e profundos com clientes que nós, da equipe de Estratégia da Arquitetura, obtemos nossas conclusões sobre as quatro áreas significativas que o setor de TI deve investir.

O ponto principal deste documento é fornecer uma visão geral sobre os desafios técnicos em uma dessas áreas, o gerenciamento de identidade e acesso, e também ajudar o leitor a compreender os problemas geralmente encontrados nesse amplo assunto.

Introdução
O I&AM (Identity and Access Management – Gerenciamento de identidade e acesso) é um termo relativamente novo que tem significados diferentes para várias pessoas. Freqüentemente, os profissionais de TI tendem a estabelecer seu significado baseados em determinados problemas relacionados à identidade e segurança que atualmente enfrentam. Por exemplo, o I&AM é entendido como um sinônimo de single sign-on, sincronização de senha, single sign-on na web, habilidades baseadas em funções e idéias semelhantes.

O objetivo principal deste artigo é fornecer ao leitor uma visão geral sucinta e abrangente sobre o que significa o I&AM. Para cumprir esse objetivo, estruturamos as informações neste documento para ajudar a responder às seguintes perguntas:
- O que é identidade digital?
- O que significa gerenciamento de identidade e acesso?
- Quais são os principais componentes tecnológicos do I&AM?
- Como os componentes do I&AM se relacionam?
- Quais são os principais desafios de arquitetura no IA&M?

Anatomia da Identidade Digital
Identificações pessoais na sociedade atual podem tomar várias formas diferentes. Alguns exemplos são carteiras de motorista, passaportes, cartões magnéticos de funcionários e cartões de clubes. Essas formas de identificação geralmente contêm informações que de alguma forma são exclusivas do proprietário, por exemplo, nomes, endereços e fotos, bem como informações sobre as autoridades que emitiram os documentos, como por exemplo, um emblema do departamento local de veículos automotores.

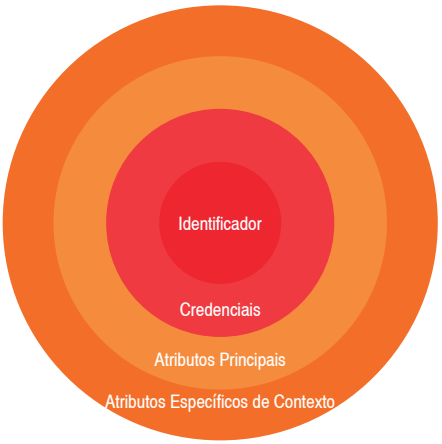
Enquanto a noção de identidades no mundo físico é muito bem compreendida, o mesmo não pode ser dito sobre a definição de identidades digitais. Para ajudar a definir o trabalho no restante da discussão neste artigo, esta seção descreve uma noção sobre a identidade digital, conforme ilustrado na Figura 1. Nossa definição de identidade digital consiste nas seguintes partes:

- *Identificador*: Informação que identifica exclusivamente o objeto dessa identidade dentro de um determinado contexto. Exemplos de identificadores são endereços de email, nomes distintos X500 e GUIDs.
- *Credenciais*: Dados privados ou públicos que podem ser usados para provar a autenticidade de uma solicitação de identidade. Por exemplo, Alice digita sua senha para provar que ela é quem diz ser. Esse mecanismo funciona porque somente o sistema de autenticação e Alice devem saber qual é a sua senha. Uma chave privada e o certificado de chave pública X509 associado é outro exemplo de credencial.
- *Atributos Principais*: Dados que

[1] O contexto define os limites nos quais a identidade é usada. Os limites podem estar relacionados a aplicações ou negócios. Por exemplo, Alice pode usar uma identidade de trabalho como o identificador Alice@WallStreetAce.com para identificar a si própria como funcionária de Wall Street (Wall Street Ace), bem como para realizar negociação de ações na NYSE. Seu status de funcionária de Wall Street e na NYSE são dois contextos de negócio diferentes em que o mesmo identificador é usado.

ajudam a descrever a identidade. Os atributos principais podem ser usados em uma variedade de contextos de aplicações ou comerciais. Por exemplo, endereços e números de telefone são atributos comuns que são usados e consultados por diferentes aplicações de negócio.
– *Atributos específicos de contexto*: Dados que ajudam a descrever a identidade, porém que são consultados e utilizados somente dentro de um contexto específico no qual a identidade é usada. Por exemplo, em uma empresa, as informações do plano de saúde de preferência do funcionário são atributos específicos de um contexto, que interessam ao fornecedor de assistência médica, mas não necessariamente ao fornecedor se serviços financeiros.

Figura 1. Anatomia da Identidade Digital



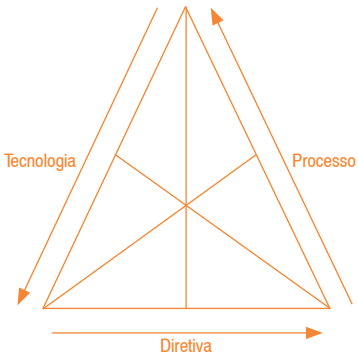
O que é Gerenciamento de Identidade e Acesso?

O Burton Group define o gerenciamento de identidade da seguinte forma: *"Gerenciamento de identidade é um conjunto de processos de negócio e uma infra-estrutura com suporte para a criação, manutenção e uso de identidades digitais".*[2]

Neste artigo, definimos o gerenciamento de identidade e acesso (I&AM) da seguinte forma: *"O gerenciamento de identidade e acesso refere-se ao processo, às tecnologias e às políticas para gerenciar identidades digitais e controlar como essas identidades podem ser usadas para acessar recursos".*

Podemos fazer algumas observações importantes com base nas definições anteriores:
– O I&AM encontra-se perto do gerenciamento do ciclo de vida ponto a ponto de identidades digitais. Uma solução de gerenciamento de identidade de classe empresarial não deve ser composta de silos isolados de tecnologias de segurança, mas, em vez disso, consistir em uma estrutura bem integrada de tecnologias que lidam com o espectro de situações em cada estágio do ciclo de vida da identidade. Falaremos mais sobre essas situações em uma seção posterior deste artigo.
– O I&AM não trata apenas de tecnologia; ao contrário, é composta de três elementos indispensáveis: diretivas, processos e tecnologias. As diretivas referem-se às restrições e aos padrões que precisam ser seguidos para atender aos requisitos de conformidade com as regulamentações e práticas comerciais recomendadas; os processos descrevem as seqüências de etapas que levam à conclusão das tarefas ou funções de negócio; as tecnologias são ferramentas automatizadas que ajudam a cumprir os objetivos de negócio de forma mais eficiente e precisa enquanto atende às limitações e orientações especificadas nas diretivas.
– As relações entre os elementos do I&AM podem ser representadas como o triângulo ilustrado na Figura 2. De interesse significativo é o fato que existe um loop de feedback que liga todos os três elementos. Os comprimentos das extremidades representam as proporções dos elementos relacionados um ao outro em um determinado sistema I&AM. A variação da proporção de um elemento irá variar à proporção de um ou mais elementos para manter a forma de triângulo com um ponto ideal (exibido como uma intersecção no triângulo).

Figura 2. Elementos essenciais do sistema do gerenciamento de identidade e acesso



– A analogia do triângulo é perfeita para descrever as relações e interações de diretivas, processos e tecnologias em um sistema I&AM íntegro. Cada organização é diferente e a combinação correta de tecnologias, diretivas e processos para uma empresa pode não ser necessariamente o equilíbrio correto para outra empresa diferente. Portanto, cada organização precisa encontrar seu equilíbrio próprio representado pela exclusividade de seu triângulo.
– O sistema I&AM da organização não permanece estático com o passar do tempo. Novas tecnologias serão introduzidas e adotadas, novos modelos e restrições de negócio irão mudar o controle e os processos corporativos para realizar atividades. Conforme mencionamos

[2] Retirado de Enterprise Identity Management: It's About the Business, Jamie Lewis, The Burton Group Directory and Security Strategies Report, v1, 2 de julho de 2003.

[3] O termo "ciclo de vida" quando usado no contexto de identidades digitais é de alguma forma inapropriado, visto que identidades digitais normalmente não são recicladas. No entanto, como o termo "gerenciamento do ciclo de vida da identidade" é defendido na comunidade de TI, iremos manter a terminologia "ciclo de vida".

anteriormente, quando um dos elementos muda, é o momento de encontrar um novo equilíbrio. Conseqüentemente, é importante compreender que o I&AM é uma jornada, e não um destino.

Estrutura do Gerenciamento de Identidade e Acesso

Conforme implícito nas seções anteriores, gerenciamento de identidade e acesso é um tópico muito amplo que abrange tanto tecnologia como outras áreas. Concentraremos o restante deste artigo nos aspectos tecnológicos de gerenciamento de identidade e acesso.

Para conter o escopo técnico deste tópico que ainda é suficientemente amplo, é útil ater-se a alguma estrutura para nossas discussões. Utilizaremos a estrutura mostrada na Figura 3, que ilustra vários componentes lógicos do I&AM para direcionar as discussões sobre o assunto.

Essa estrutura particular destacou três "compartimentos" fundamentais dos componentes tecnológicos:

- Gerenciamento do ciclo de vida da identidade
- Gerenciamento de acesso
- Serviços de diretório

Figura 3. Componentes Lógicos do I&AM

Os componentes nesses compartimentos lógicos são usados para atender a um conjunto de requisitos recorrentes nas soluções de gerenciamento de identidade. Descreveremos as funções que esses componentes desempenham nas próximas seções.

Serviços de Diretório

Conforme mencionado anteriormente, a identidade digital consiste em alguns tipos lógicos de dados – o identificador, as credenciais e os atributos. Esses dados precisam ser armazenados e organizados de forma segura. Os serviços de diretório fornecem a infra-estrutura para atender a tais necessidades. Habilidades e diretivas de segurança freqüentemente controlam o acesso e o uso de aplicações de negócio e a infra-estrutura de computação dentro de uma organização. As habilidades são direitos e privilégios associados a indivíduos ou grupos. As diretivas de segurança referem-se às normas e restrições sob as quais os recursos de computação de TI operam. Uma diretiva de complexidade de senha é um exemplo de diretiva de segurança. Outro exemplo é a configuração de confiança de uma aplicação de negócio que pode descrever terceiros em quem as aplicações confiam para ajudar a autenticar e identificar os usuários da aplicação. Como as identidades digitais, as diretivas de segurança e as habilidades precisam ser armazenadas, gerenciadas adequadamente e descobertas. Em muitos casos, os serviços de diretório fornecem uma boa base para atender a esses requisitos.

Gerenciamento de Acesso

O gerenciamento de acesso refere-se ao processo de controlar e conceder acesso para atender às solicitações de recursos. Esse processo geralmente é concluído por meio de uma seqüência de ações de autenticação, autorização e auditoria. A autenticação é o processo pelo qual as solicitações de identidades são comprovadas. A autorização é a determinação se alguma identidade tem permissão para executar ações

ou acessar recursos. A auditoria é o processo de contabilização para registrar eventos de segurança ocorridos. Unidas, a autenticação, a autorização e a auditoria são também comumente consideradas o padrão ouro de segurança. (O motivo por trás disso origina-se do símbolo periódico para ouro, "Au" que é o prefixo dos três processos).

Existem vários problemas técnicos que arquitetos de soluções podem encontrar ao criar e integrar mecanismos de autenticação, autorização e auditoria na arquitetura de aplicações:
– Single Sign-On
– Confiança e Federação
– Habilidades do Usuário
– Auditoria

Descreveremos esses desafios e suas soluções com mais detalhes posteriormente neste documento.

Gerenciamento do Ciclo de Vida da Identidade

O ciclo de vida de uma identidade digital pode ser estruturado em estágios semelhantes ao ciclo de vida de seres vivos:
– Criação
– Utilização
– Término

Cada estágio no ciclo de vida da identidade apresenta situações em que existem candidatos para o gerenciamento automatizado. Por exemplo, durante a criação de uma identidade digital, os dados da identidade precisam ser propagados e inicializados em sistemas de identidade. Em outras situações, as habilidades de identidade talvez precisem ser ampliadas quando o usuário representado pela identidade receber alguma promoção de cargo. Finalmente, quando a identidade digital não estiver mais em atividade, talvez seu status precise ser alterado ou talvez seja necessário que a identidade seja excluída do armazenamento de dados.

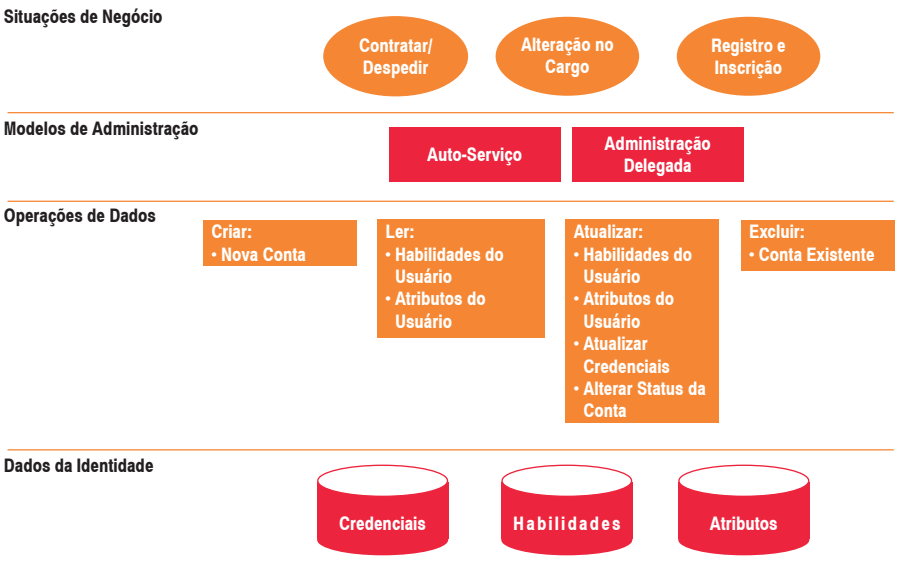
Todos os eventos durante o ciclo de vida de uma identidade digital precisam ser gerenciados de forma segurança, eficiente e precisa, que é exatamente o que significa o gerenciamento do ciclo de vida da identidade.

Os requisitos para o gerenciamento do ciclo de vida da identidade podem ser discutidos em vários níveis, conforme representado na Figura 4. Os tipos de dados que precisam ser gerenciados são exibidos no nível dos dados da identidade. Com base em nossas definições de identidade digital anteriores, os dados relevantes incluem credenciais, como senhas e certificados, e atributos de usuário, como nome, endereço e números de telefone. Além das credenciais e dos atributos, existem também os dados de habilidades do usuário para gerenciar. Essas habilidades são descritas com mais detalhes posteriormente; neste momento, as habilidades devem ser consideradas como os direitos e privilégios associados às identidades.

Passando para o próximo nível na ilustração, os requisitos listados refletem os tipos de operações que podem ser realizadas nos dados da identidade. Criar, Ler, Atualizar e Excluir (CRUD) são primitivas de operação de dados inventadas pela comunidade de banco de dados. Reutilizamos essas primitivas aqui, pois fornecem uma forma muito cômoda para classificar os tipos de operações do gerenciamento de identidades. Por exemplo, podemos classificar alterações em status de contas, habilidades e credenciais sob a primitiva de dados Atualizar.

O próximo nível na ilustração mostra dois modelos de administração do ciclo de vida da identidade: auto-serviço e modelo delegado. Nas organizações de TI tradicionais, tarefas de administração de computador são realizadas por um grupo centralizado de administradores de sistema. Com o passar do tempo, as organizações perceberam que podem existir bons motivos econômicos e de negócio para permitir outros tipos de modelos de

Figura 4. Níveis dos Requisitos do Gerenciamento do Ciclo de Vida da Identidade



administração. Por exemplo, com freqüência é mais eficiente em termos de custo que os indivíduos possam atualizar por conta própria alguns de seus atributos pessoais, como endereço e número de telefone. O modelo de administração de auto-serviço permite esse poder individual. O ponto intermediário entre o auto-serviço e o modelo de administração centralizada é a administração delegada. No modelo delegado, as responsabilidades da administração do ciclo de vida da identidade são compartilhadas entre grupos descentralizados de administradores. Os critérios comuns usados para determinar o escopo da delegação são funções de administração e da estrutura da organização. Um exemplo de administração delegada baseada na estrutura da organização é a hierarquia dos administradores de nível empresarial, de unidade e departamento em grandes organizações.

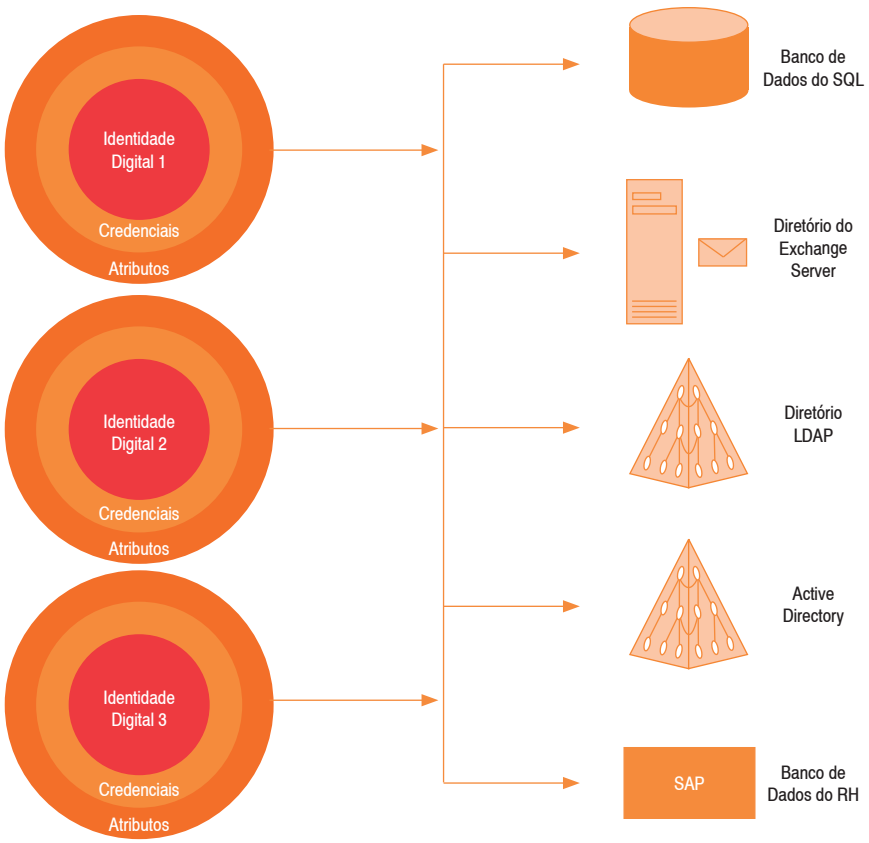
Os modelos de administração do ciclo de vida demonstrados anteriormente podem ser usados para dar suporte a uma variedade de situações de negócio, algumas das quais estão listadas na *Figura 4*. Por exemplo, freqüentemente é necessário que contas sejam criadas e fornecidas para novos funcionários. Por outro lado, quando algum funcionário não trabalha mais para a empresa, talvez o status da conta existente precise ser alterado. Situações de alteração de cargo podem causar vários impactos nas identidades digitais. Por exemplo, quando Bob receber uma promoção, talvez seu cargo precise ser alterado e suas habilidades estendidas.

Agora que compreendemos melhor os requisitos de gerenciamento do ciclo de vida da identidade, estamos prontos para nos aprofundar nos desafios envolvidos em atender a esses requisitos. A ilustração na *Figura 5* destaca o fato de que um usuário típico em uma empresa normalmente precisa lidar com várias identidades digitais que podem ser armazenadas e gerenciadas de forma independente. Essa situação atual ocorre devido à evolução constante que toda organização de negócio enfrenta. Eventos como fusões e aquisições podem introduzir sistemas incompatíveis em infra-estruturas de TI existentes; e os requisitos de negócio em evolução podem ter sido atendidos por meio de aplicações de terceiros que não estão bem integradas com as existentes.

Alguém pode presumir que seria muito mais fácil para as empresas substituir os sistemas atuais e começar novamente. No entanto, essa solução dificilmente é uma opção viável. Temos que reconhecer que os sistemas de identidade existentes estão disponíveis há um longo tempo, o que nos leva a encontrar outras soluções para resolver dois problemas críticos que surgem do gerenciamento de dados em sistemas de identidade distintos:

1. Duplicação de informações
   As informações da identidade são duplicadas com freqüência em vários sistemas. Por exemplo, atributos como endereços e números de telefone são freqüentemente armazenados e gerenciados em mais de um sistema no ambiente. Quando algum dado de identidade é duplicado, pode ficar facilmente fora de sincronia caso atualizações sejam feitas em um sistema, mas não nos outros.

Figura 5. Situação atual: vários sistemas de identidade na empresa



2. Falta de integração
   A visão completa dos atributos, das credenciais e dos privilégios de um determinado usuário geralmente é distribuída entre vários sistemas de identidade. Por exemplos, para um determinado funcionário, as informações relacionadas a recursos humanos podem estar contidas em sistemas SAP de RH, a conta de acesso de rede no Active Directory e os privilégios de aplicações legado armazenadas em mainframes. Várias situações de gerenciamento de ciclo de vida de identidade exigem que as informações de identidade sejam introduzidas e retiradas de vários sistemas diferentes.

Iremos nos referir aos problemas descritos anteriormente como o desafio de agregação de identidade, que será descrito com detalhes em uma seção posterior.

Desafios no Gerenciamento de Identidade e Acesso

Single Sign-On
Um usuário empresarial típico precisa fazer login várias vezes para obter acesso a várias aplicações que utiliza em seus trabalhos. Do ponto de vista do usuário, vários logins e a necessidade de lembrar-se de várias senhas é a principal causa de experiências ruins com aplicações. Do ponto de vista do gerenciamento, incidentes de esquecimento de senhas definitivamente aumentam custos de gerenciamento e, quando combinados com hábitos incorretos de gerenciamento de senha de usuário (como escrever as senhas em notas auto-adesivas), geralmente podem levar a mais oportunidades de vulnerabilidade de segurança. Devido aos problemas aparentemente insolúveis apresentados nas identidades, o conceito de SSO (Single Sign-On), a capacidade de fazer login uma vez e obter acesso a vários sistemas, tornou-se o "Santo Graal" dos projetos de gerenciamento de identidades.

Soluções de Single Sign-On
De forma ampla, existem cinco classes de soluções SSO. Nenhum tipo de solução é correto para cada situação de aplicação. A melhor solução depende grandemente de vários fatores, como onde as aplicações que requerem o SSO encontram-se hospedadas, as limitações impostas pela infra-estrutura (exemplo, restrições de firewall) e a capacidade de modificar as aplicações. Estas são as cinco categorias das soluções SSO:
1. SSO na Web
2. Sign-On Integrado no Sistema Operacional
3. Sign-On Federado
4. Mapeamento de Identidade e Credencial
5. Sincronização de Senha

As soluções de SSO na web foram criadas para atender aos requisitos de sign-on em aplicações web. Nessas soluções, usuários de navegadores não autenticados são redirecionados a sites de login para introduzir as identificações e credenciais de usuário. Após a autenticação bem-sucedida, cookies HTTP são emitidos e usados por aplicações web para validar as sessões de usuário autenticadas. O Microsoft Passport é um exemplo de solução SSO na web.

Sign-on integrado em sistemas operacionais refere-se a módulos de autenticação e interfaces criadas no sistema operacional. O subsistema de segurança do Windows fornece tal recurso por meio de módulos de sistema como o LSA (Local Security Authority) e o SSP (Security Specific Providers). SSPI refere-se às interfaces de programação nesses SSPs. Aplicações de desktop que usam as APIs SSPI para autenticação de usuário podem aproveitar o login da área de trabalho do Windows para ajudar a obter o SSO na aplicação.

O GSSAPI em várias implementações UNIX também fornece a mesma funcionalidade do SSO em aplicações.

O sign-on federado requer que as infra-estruturas de autenticação de aplicações compreendam as relações de confiança e interoperem por meio de protocolos padrão. Kerberos e o novo Serviço de Federação do Active Directory são exemplos de tecnologia de federação. Sign-on federado significa que a responsabilidade de autenticação é delegada a alguém confiável. Usuários de aplicações não precisam receber solicitações para fazer login novamente contanto que o usuário tenha sido autenticado por um componente de infra-estrutura de autenticação federado (ou seja, confiável).

Soluções de mapeamento de identidade e credenciais geralmente usam caches de credenciais para manter o controle das identidades e credenciais a serem utilizadas para acessar as listas correspondentes de sites de aplicações. O cache pode ser atualizado manualmente ou automaticamente quando a credencial (por exemplo, senha) for alterada. As aplicações existentes podem ou não precisar de alterações para usar as soluções de mapeamento de identidade. Quando a aplicação não puder ser modificada, um agente de software pode ser instalado par monitorar os eventos de login na aplicação. Quando o agente detectar tais eventos, ele localiza a credencial do usuário no cache e a introduz automaticamente no prompt de login na aplicação.

A técnica de sincronização de senha é usada para sincronizar senhas em bancos de dados de credenciais de aplicações de forma que usuários e aplicações não precisem gerenciar várias alterações em senhas. A sincronização de senha como uma tecnologia silo não fornece realmente um single sign-on, mas resulta em algumas comodidades das quais as aplicações podem tirar proveito. Por exemplo, com a sincronização de senha, uma aplicação de camada intermediária pode presumir que a

senha de um usuário de aplicação é a mesma nos vários sistemas que precisa acessar para que a aplicação não precise procurar por senhas diferentes para usar ao acessar recursos nesses sistemas.

Gerenciamento de Habilidades

O gerenciamento de habilidades refere-se ao conjunto de tecnologias usado para conceder e invocar direitos de acesso e privilégios para identidades. Está estreitamente associado à autorização, que é o processo real de impor regras de acesso, diretivas e restrições que estão associadas a funções e dados de negócio.

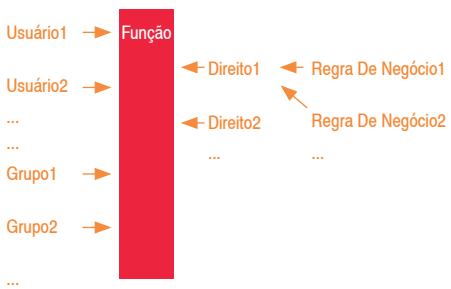
As aplicações empresariais da atualidade freqüentemente usam uma combinação de autorização baseada em função e diretivas baseadas em regras para determinar o que uma dada identidade pode ou não fazer.

Dentro de uma aplicação distribuída com n camadas, decisões de acesso podem ser tomadas em qualquer camada da arquitetura da aplicação. Por exemplo, a camada de apresentação pode apresentar somente escolhas de interface de usuário que o usuário está autorizado a fazer. Na camada de serviço da arquitetura, o serviço pode verificar se o usuário respeita a condição de autorização para solicitar o serviço. Por exemplo, somente usuários com a função de gerente podem chamar o serviço "Aprovação de Empréstimo". Por trás da camada lógica de negócio, pode haver a necessidade de decisões refinadas de diretivas de negócio como "Esta solicitação foi feita durante o horário comercial". Na camada de dados, o procedimento armazenado no banco de dados pode filtrar dados devolvidos sobre a relação entre a identidade do solicitador do serviço e o dado solicitado.

Dada a utilidade e o freqüente uso cruzado de esquemas de autorização baseado em regras e funções, nem sempre é claro para arquitetos de aplicações como modelar uma estrutura de gerenciamento de habilidades que integre ambos os esquemas de forma exata. Muitas empresas possuem mecanismos personalizados separados para fazer ambos.

A *Figura 6* ilustra uma representação de como ambos os esquemas podem ser combinados e integrados. Nessa representação, podemos prever uma definição de função (que normalmente reflete a responsabilidade do cargo) com dois conjuntos de propriedades. Um conjunto de propriedades contém as identidades das pessoas ou sistemas que se encontram em uma determinada função. Por exemplo, Alice, Bob e Charlie podem ser atribuídos com a função Gerente. O segundo conjunto de propriedades contém o conjunto de direitos que uma determinada função possui. Direitos podem representar funções ou ações de negócio em recursos de computação. Por exemplo, transferência de fundos define uma função de negócio e leitura de arquivo refere-se a uma operação em um recurso de computação. Além disso, podemos atribuir um conjunto de instruções condicionais (regras de negócio) para cada direito. Por exemplo, o direito transferência de fundos pode ter uma instrução condicional para permitir a ação se o momento atual estiver dentro do horário comercial. Observe que a instrução condicional pode basear sua decisão nos dados dinâmicos de entrada que somente podem ser determinados no runtime da aplicação.

Figura 6. Autorização baseada em funções e regras integradas



Também é importante que a maioria das organizações tenha uma visão consolidada de todos os direitos que uma determinada identidade possui. Para atender a esse requisito, aplicações de gerenciamento de habilidades normalmente aproveitam um armazenamento de diretivas centralizado para ajudar a facilitar o gerenciamento centralizado e relatórios de direitos dos usuários.

Agregação da Identidade

Sistemas de TI empresariais evoluem de forma orgânica durante o histórico de uma organização. Isso é devido a motivos como fusões e aquisições, ou preferências e mudanças dos líderes de TI. As conseqüências disso são freqüentemente manifestadas por meio de uma miscelânea de sistemas de TI desconectados com artefatos de arquitetura indesejados. Sistemas relacionados a identidades não são exceções a tais evoluções no setor de TI.

Freqüentemente, a empresa terá não só um sistema de identidades, mas vários, cada um com funções de negócio diferentes, porém armazenando dados relacionados e duplicados. Aplicações que precisam integrar-se a essas funções de negócio são então forçadas a se enquadrar às diferenças e sincronizar as duplicações.

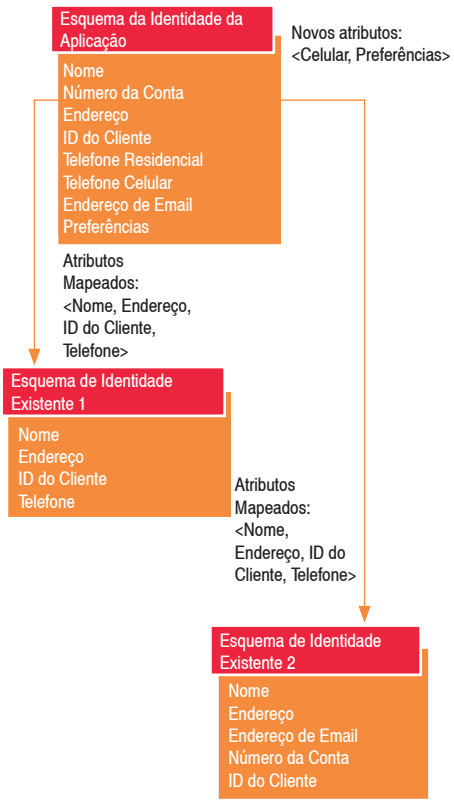
Por exemplo, uma aplicação bancária de serviços ao cliente pode precisar obter informação de um cliente em um banco de dados IBM DB2, um banco de dados Oracle e um CRM local. Nesse caso, o conceito da aplicação com relação ao "Cliente" é definido por três sistemas diferentes. Atributos que descrevem o cliente, como nome, endereço e número da previdência social, podem ser armazenados e duplicados em vários sistemas. Por outro lado, dados

não duplicados, como os produtos financeiros que o cliente adquiriu e o saldo bancário do cliente, podem ser mantidos em sistemas separados. A aplicação precisará agregar esses dados de sistemas deferentes para obter a visão necessária.

A passagem dos dados de identidade subjacentes para um gigantesco sistema de identidades pode parecer uma resposta óbvia para o problema. No entanto, existem muitos problemas no mundo real (por exemplo, o risco de quebrar aplicações legadas) que impedem que tais soluções sejam amplamente adotadas.

A agregação de identidade, portanto,

Figura 7. Reconciliação de esquemas de identidade



refere-se ao conjunto de tecnologias que ajudam aplicações a agregar informações de identidade de diferentes sistemas, reduzindo a complexidade da reconciliação, sincronização e integração de dados.

Existem vários desafios técnicos que as tecnologias de agregação de identidades podem ajudar a solucionar:

- Manutenção de relações para transformações de dados.
- Otimização de operações CRUD de dados.
- Sincronização de dados.

As próximas subseções fornecem uma visão geral desses problemas de design.

(a) Manutenção de Relações para Transformações de Dados

Uma solução de agregação de identidade pode fornecer vários benefícios para aplicações com visões distintas de informações de identidade que manipulam dados em sistemas diferentes. O primeiro benefício envolve o fornecimento de aplicações com uma visão consolidada ou agregada dos dados nos sistemas individuais. Para transformar e representar dados em visões diferentes, a solução de agregação de identidade precisa manter metadados que descrevem o esquema representando a visão consolidada e suas relações com esquemas de dados em vários sistemas de identidade.

Vejamos um exemplo específico de aplicação de gerenciamento de ciclo de vida da identidade que gerencia dados em vários sistemas existentes. A visão consolidada da aplicação de gerenciamento é representada por um novo esquema que é composto de atributos de identidade novos e existentes.

A *Figura 7* ilustra um exemplo em que um novo esquema de identidade é definido para a aplicação. O novo esquema refere-se aos atributos em dois esquemas de identidade existentes e também define novos que não são especificados no momento (Número da Conta, Telefone Celular e Preferências).

Para que as aplicações consultem e atualizem os dados em armazenamentos existentes, precisaremos manter relações de dados entre os atributos definidos no novo esquema e os atributos correspondentes nos esquemas existentes. Por exemplo, as seguintes relações precisarão ser mantidas:

Referência – Uma referência trata de informações que identificam de forma não ambígua uma instância de dados, conforme representado por um esquema de dados particular. Por exemplo, o atributo "ID de Cliente" permite uma instância de dados, conforme representado pelo esquema da aplicação na *Figura 7* para encontrar sua instância de dados correspondente, conforme representado pelo esquema existente. Diferentes esquemas podem usar diferentes referências para identificar suas próprias instâncias de dados.

Propriedade – Um atributo de dados pode ser definido em mais de um esquema existente. Usando o mesmo exemplo exibido na *Figura 7* podemos ver novamente que os atributos "Nome" e "Endereço" estão definidos em ambos os esquemas existentes. No caso de conflitos de dados, a aplicação precisa saber qual versão possui a cópia com autoridade a ser mantida. Além

disso, pode haver situações em que um atributo obtenha seu valor de uma lista de prioridades de proprietários. Em outras situações, quando o valor para um atributo não estiver presente na primeira fonte com autoridade, o serviço de agregação deve consultar o sistema seguinte na lista de prioridades de proprietários.

Mapeamento de Atributos
– Os atributos definidos em vários armazenamentos de dados podem apresentar o mesmo significado semântico, mas possuem representações sintáticas iguais ou diferentes. Ao consultar ou atualizar um atributo de dados, o serviço de agregação da identidade precisa saber quais atributos são equivalentes semanticamente. Por exemplo, embora a ID do cliente seja definida em todos os três esquemas, ela possui nomes diferentes, como CustID, no esquema de identidade 2. Quando a aplicação executa uma atualização no número da ID do cliente da identidade, o sistema também deve atualizar o atributo CustID para a instância de dados representada pelo esquema 2.

(b) Otimização de Operações CRUD de Dados
Conforme identificado anteriormente, o CRUD é um acrônimo de banco de dados para Create, Read, Update e Delete (Criar, Ler, Atualizar e Excluir). O CRUD define as operações primitivas básicas para manipular dados. O motivo pelo qual o CRUD é considerado um desafio técnico é porque o desempenho para concluir uma atividade relacionada à agregação que envolva operações CRUD entre vários sistemas back-end de dados pode variar significativamente dependendo das relações dos dados.

No melhor dos casos, as operações CRUD podem ser paralelizadas. Isso é verdade em situações nas quais as instâncias de dados podem ser resolvidas usando a mesma referência, e a referência de dados sempre resolve uma instância exclusiva. Por exemplo, se o número da previdência social for a única chave usada para consultar e agregar dados entre armazenamentos de dados, essa consulta particular pode ser emitida em paralelo.

No pior dos casos, as operações CRUD são serializadas entre armazenamentos de dados. A operação serializada é comum em situações em que as referências usadas para resolver instâncias de dados possuem dependências em outras instâncias de dados. Como uma ilustração simples, suponhamos que precisemos agregar dados de bancos de dados de RH e Benefícios e que as referências de instâncias são a ID do funcionário e a ID de previdência social, respectivamente. Se a única chave inicial que possuímos para a consulta for a ID do funcionário, e a ID da previdência social puder ser obtida somente pelo banco de dados de RH, então precisaremos serializar a consulta na seguinte ordem:

1. Consulta ao banco de dados do RH usando a ID do funcionário como a chave.
2. Extração do número de previdência social com base nos resultados da consulta anterior.
3. Consulta do banco de dados de benefícios.
4. Agregação dos resultados da consulta.

A replicação é uma técnica comum usada para resolver a degradação do desempenho devido a operações CRUD em armazenamentos de dados. Para solucionar esse problema de desempenho, uma parte dos atributos de dados back-end pode ser replicada para um armazenamento mantido pelo serviço de agregação da identidade. Além disso, a cópia local do dado replicado pode ser desnormalizada para ajudar a melhorar o desempenho do CRUD.

(c) Sincronização de Dados
A sincronização de dados é necessária nos casos em que uma ou ambas das seguintes condições forem verdadeiras:

– Existem atributos de identidade duplicados em vários armazenamentos back-end.
– Os dados são replicados em um armazenamento agregador de identidade intermediário.

Entretanto, o uso da sincronização de dados pode também introduzir outros problemas de design:
– Resolução de conflito de dados. Nos casos em que os dados podem ser atualizados de mais de uma fonte, com freqüência é fácil introduzir dados conflitantes. Algumas práticas comuns que ajudam a atenuar esses casos são as seguintes:
– Atribuição de prioridade de propriedade de dados de forma que, no caso de conflitos, a autoridade atribuída seja utilizada.
– Durante algum conflito, o último autor vence.
– Ativação da sincronização. Algumas abordagens comuns são:
  • Atualizações programadas.
  • Evento baseado em notificação.

Confiança e Federação
Conforme mencionado na seção sobre single sign-on, a federação oferece uma forma de solução de single sign-on. No entanto, a federação é mais que apenas um single sign-on. A federação implica a delegação de responsabilidades honradas por meio de relações de confiança entre as parte federadas. A autenticação é apenas uma forma de responsabilidade delegada. A autorização, o gerenciamento de perfis, os serviços de pseudônimo e a cobrança são outras formas de funções relacionadas à identidade que podem ser delegadas a alguém confiável.

Existem três elementos de tecnologia que são fundamentais ao conceito de federação:
– Um protocolo de federação que permite a comunicação entre as partes.
– Uma infra-estrutura de confiança flexível que oferece suporte a uma

variedade de modelos de confiança.
– Uma estrutura de gerenciamento de diretivas extensível que oferece suporte a diferentes requisitos de controle.

Os protocolos de federação são as "linguagens" usadas pelas partes federadas para comunicarem-se umas com as outras.- Como a federação implica que a responsabilidade seja delegada e executada por uma parte diferente, o protocolo deve permitir que indivíduos obtenham "recursos" – essencialmente solicitações invioláveis indicando que uma determinada identidade concluiu com êxito alguma ação ou possui habilidades para uma variedade de privilégios. Por exemplo, no caso de um sign-on federado, uma identidade de autenticação obtém uma capacidade que prova que o indivíduo autenticou com êxito por meio de um serviço de autenticação aprovado.

Em um mundo de negócio complexo, é possível possuir esquemas de confiança relativamente complexos que envolvam várias partes comerciais. A tecnologia de federação deve ter a capacidade de capturar a essência dessas relações de confiança reais em modelos de confiança simples de compreender, porém avançados, que irão ajudar a permitir várias situações de negócio. A seguir encontram-se alguns modelos de confiança comuns, ilustrados na *Figura 8*:
– Hub-and-spoke
– Hierárquico
– Rede de confiança ponto-a-ponto

O modelo hub-and-spoke é o mais fácil de compreender. Nesse modelo, existe um broker central que é confiado diretamente pelas partes federadas. A União Européia é um exemplo desse modelo de federação, em que os respectivos países confiam diretamente no organismo da UE para fornecer orientações econômicas comuns e oportunidades de comércio às partes federadas.

No modelo hierárquico, duas partes possuem uma relação de confiança indireta se ambas tiverem um caminho de confiança em suas respectivas ramificações, na árvore hierárquica, a uma autoridade raiz comum. O sistema político americano demonstra esse modelo de confiança. Esse sistema possui organismos políticos municipais, de condados, estaduais e federais, cada um existindo em vários níveis da hierarquia.

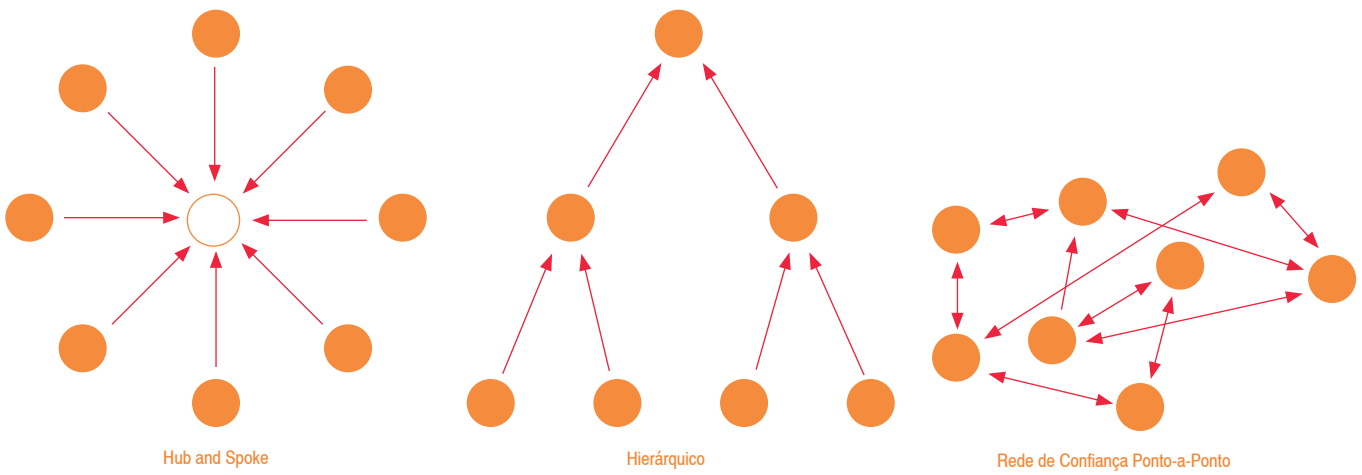
O modelo ponto-a-ponto representa uma coleção de relações de confiança diretas ad-hoc. As relações pessoais entre amigos no mundo físico são bons exemplos desse modelo de confiança.

Observe que também é possível estender e formar novas redes de federação que consistem em diferentes modelos de confiança, ou modelo híbridos.

Uma estrutura básica de gerenciamento de diretivas deve permitir a criação, exclusão, modificação e descoberta de diretivas. Para promover sistemas de federação que permitam a rápida integração de novos modelos e parcerias de negócio, a estrutura de gerenciamento de diretivas também deve ser extensível para refletir a dinâmica do ambiente que pretende oferecer suporte.

Alguns exemplos de diretivas de aplicações que são relevantes no mundo federado são:
– Emissor confiável de recursos relacionados a identidades.
– Os tipos de recursos necessários para solicitar a operação da aplicação.

Figura 8. Modelos de confiança comuns



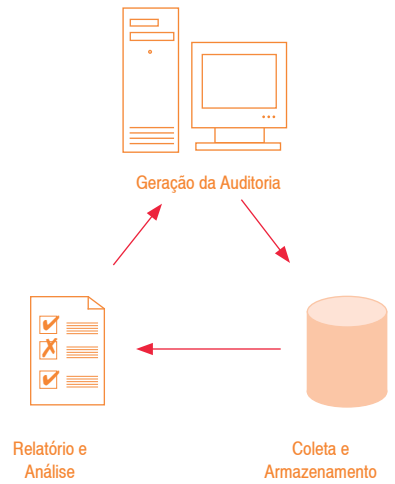
"A federação implica a delegação de responsabilidades honradas por meio de relações de confiança entre as partes federadas."

– Os tipos de informações de identidade que a aplicação espera nos recursos.
– Os tipos de privilégios que a identidade deve demonstrar para solicitar um serviço.

Auditoria

A auditoria no contexto do I&AM trata de manter registros de "quem fez o que e quando" dentro da infra-estrutura de TI. As regulamentações federais, como a Lei Sarbanes-Oxley, são os principais condutores dos requisitos de auditoria relacionados a identidades.

Figura 9. Processo de auditoria de TI



(a) Processo de Auditoria de TI

O processo de auditoria de TI normalmente envolve as seguintes fases, conforme ilustrado na Figura 9:
– Geração da auditoria
– Coleta e armazenamento de dados
– Análise e feedback

As etapas da auditoria podem ser geradas por diferentes infra-estruturas e componentes de aplicações para diferentes finalidades. Por exemplo, firewalls e servidores VPN podem gerar eventos para ajudar a detectar violações externas; componentes intermediários podem gerar dados de instrumentação para ajudar a detectar anomalias no desempenho; e aplicações de negócio podem produzir dados de auditoria para ajudar a depurar ou cumprir os requisitos de auditoria de regulamentação.

Depois que a auditoria for produzida, será necessário coletá-la e armazená-la. Existem dois modelos principais a serem considerados aqui: distribuído e centralizado. No modelo distribuído, os dados da auditoria normalmente permanecem no sistema onde o dado foi gerado. Com a abordagem centralizada, o dado é enviado a um local de armazenamento de dados e coleta central.

Uma vez que os dados forem coletados, poderão ser processados e analisados automática ou manualmente. A análise de auditoria foi criada para levar a conclusões sobre quais ações corretivas, se houver, são necessárias para melhorar sistemas e processos de TI.

(b) Considerações sobre o Design de Sistemas de Auditoria

Com base no processo típico de auditoria, descrito nas seções anteriores, existem várias considerações importantes ao design de sistemas de auditoria:
– Localidade da geração e do armazenamento da auditoria
– Separação da função do auditor
– Fluxo dos eventos auditados

Depois que as etapas de auditoria forem geradas, os dados da auditoria poderão ser armazenados localmente no mesmo sistema ou transferidos para outro local de armazenamento. Essa consideração é importante sob a perspectiva de segurança, visto que os dados de auditoria podem ser alterados com facilidade se forem armazenados no mesmo sistema que os gerou. Esse ponto pode ser ilustrado por um exemplo simples: no caso de um sistema comprometido, o invasor pode modificar as etapas de auditoria no sistema local para ocultar a tentativa de invasão. Portanto, para obter mais segurança contra tais ataques, pode-se desejar que o sistema de auditoria armazene os dados separadamente em um equipamento remoto.

A separação de funções é uma prática recomendada para ajudar a minimizar a ocorrência de atividades ilegais que resultam de ações que possam contornar verificações de responsabilidade. Por exemplo, um gerente de compras corporativas não pode aprovar solicitações de compra. No campo de auditoria de TI, é uma prática comum separar a função do administrador do sistema da função do auditor. Isso impede que o administrador do sistema, que geralmente possui um "status superior" sobre computadores, encubra eventos de auditoria de atividades não autorizadas.

Em um modelo de design distribuído, em que os dados da auditoria são gerados e armazenados em sistemas diferentes, somente a permissão saída de dados dos sistemas de auditoria pode adicionar outro nível de segurança. Essa medida de prevenção reduz as possibilidades de que dados de auditoria violados substituam as etapas reais.

Além disso, espera-se que as infra-estruturas de auditoria de identidade sejam:
– Eficientes (normalmente indica um canal de comunicação assíncrono baseado em mensagens).
– Disponíveis (normalmente indica distribuída, agrupada e tolerante a falhas).
– Precisas (para manter registros e rastros precisos).
– Não repudiada (pode ser admitida em tribunais como evidência, normalmente indica que os registros precisam ser assinados digitalmente).

Conclusões
As organizações são compostas de pessoas e sistemas, representadas dentro de sistemas de TI como identidades digitais. Tudo que ocorre nos negócios é conseqüência de ações iniciadas por essas identidades e para elas. Sem as identidades (mesmo as anônimas) não existiriam atividades e os negócios seriam construções sem vida.

Nesse nível mais alto, SOA pode ser vista como uma forma de organizar a infra-estrutura de TI comercial que executa e gerencia as atividades de negócio. Portanto, as empresas que buscam tornar SOA uma realidade devem descobrir como resolver os desafios de gerenciamento de identidade e acesso que enfrentam na atualidade. Nós fornecemos uma visão geral de algumas áreas tecnológicas principais:

– O alcance de single sign-on de aplicações e usuários.
– Os dados de agregação, transformação, sincronização e provisão de identidade de vários sistemas.
– O gerenciamento do acesso às funções e aos dados de negócio usando funções e regras.
– A federação com parceiros de negócio.
– As atividades relacionadas à auditoria de identidade.

Também esperamos que as discussões técnicas sobre os desafios tenham ajudado os leitores a ter alguma apreciação dos produtos e soluções mencionados neste espaço.

Nossa conclusão final:
O gerenciamento de identidade e acesso é o alicerce para realizar SOA. Mostre-me uma empresa que se considere "bem-SOAcedida" e mostrarei uma empresa que tem um bom controle de seu gerenciamento de identidade e acesso.

Referências

1. *SOA Challenges: Entity Aggregation*, Ramkumar Kothandaraman, Centro de Arquitetura .NET, maio de 2004 (URL: http://msdn.microsoft.com/architecture/default.aspx?pull=/library/en-us/dnbda/html/dngrfsoachallengesentityaggregation.asp)
2. *Enterprise Identity Management: It's About the Business*, Jamie Lewis, The Burton Group Directory and Security Strategies Report, v1, 2 de julho de 2003
3. *Microsoft Identity and Access Management Series*, Microsoft Corporation, 14 de maio de 2004 (URL: http://www.microsoft.com/downloads/details.aspx?FamilyId=794571E9-0926-4C59-BFA9-B4BFE54D8DD8&displaylang=en)
4. *Enterprise Identity Management: Essential SOA Prerequisite Zapflash*, Jason Bloomberg, Zapflash, 19 de junho de 2003 (URL: http://www.zapthink.com/report.html?id=ZapFlash-06192003)

Frederick Chong
Arquiteto de Soluções
Microsoft Corporation
fredch@microsoft.com

Frederick Chong é arquiteto de soluções na Equipe de Estratégia de Arquitetura na Microsoft, onde oferece orientação sobre aplicações de integração com tecnologias de gerenciamento de identidade e acesso. Ele descobriu seu interesse na segurança enquanto criava o protótipo de um protocolo de dinheiro eletrônico no grupo de segurança de rede na IBM T J Watson Research Center. Desde então, tem implementado recursos de segurança e protocolos de reforço de licenças nas equipes de produtos Microsoft e colaborado com vários clientes empresariais para projetar e implementar uma variedade de soluções de segurança, incluindo single sign-on, SSL-VPN e protocolos de segurança de serviços web.

Business Patterns para Uso em Engenharia de Software – Parte 2

Por Philip Teale, Microsoft Corporation e Robert Jarvis, SA Ltd

Introdução
Este é o segundo de uma série de dois artigos. A finalidade destes artigos é explorar se os business patterns podem ser definidos em uma forma estruturada e, se isso for possível, se esses padrões seriam de valor para aqueles que criam sistemas de software com suporte aos negócios. Nosso ponto de vista é que esse valor realmente existe.

O primeiro artigo foi publicado no JOURNAL2 e explorou como definir business patterns que podem ser úteis para engenheiros de software. O artigo 1 define o cenário para este artigo e deve ser lido antes.

No primeiro artigo utilizamos a Estrutura de Arquitetura Empresarial chamada SAM2 para analisar a oportunidade e concluímos que tais business patterns descreverão:

– As funções de negócio que serão suportadas.

– Os dados necessários para oferecer suporte às funções descritas.

– Os componentes de negócio que são as representações de TI dos dados e das funções de que o negócio precisa.

– Opcionalmente, a infra-estrutura necessária para dar suporte às funções, aos dados e aos componentes. Isso é necessário em empresas altamente distribuídas ou aquelas compostas de divisões ou unidades com diversos ambientes técnicos ou operacionais.

Além disso, definimos os business patterns que descrevem as principais relações entre essas dimensões.

Neste segundo artigo, descrevemos como desenvolver business patterns baseados em funções, dados e componentes de negócio. Também mostraremos como podem ser usados para projetar sistemas de software.

Resumo do Artigo 2
Neste artigo usaremos um exemplo realista, porém simplificado, para

Diferença entre patterns e sistemas
Caso não tenha tido muita experiência com patterns, pode observar os exemplos contidos neste artigo e considerá-los como um desenvolvimento de sistema em vez de um pattern. Isso ocorre porque, para a engenharia de software, o pattern é o ponto de partida no caminho para desenvolver parte de algum sistema. O pattern direciona-o durante parte do caminho ao objetivo final, começando a partir de um local confiável e conhecido. Parece familiar, mas não é uma solução em si própria – é uma abstração da solução que precisa ser elaborada para criar a solução que atenda às suas necessidades. A parte complicada na criação de patterns é obter o nível de abstração correto. Pouca abstração restringe a utilidade do pattern, por ser muito específica. No entanto, muita abstração também limita a utilidade, visto que fornece pouquíssimo valor prático.

Considere hoje uma situação comum em que se criou um serviço chamado "corretor de mensagens" para fornecer o roteamento e a transformação de mensagens entre uma variedade de serviços TI. Um pattern com alta abstração pode ser exatamente o que foi mencionado – "se desejar fornecer roteamento e transformação de mensagens entre uma variedade de serviços de TI utilize um corretor de mensagens". Qual foi sua utilidade? Bem, faz com que você pense da maneira correta, então possui alguma utilidade, mas não fornece muita ajuda prática.

Por outro lado, depois de criar um corretor de mensagens para manipular os serviços de TI exigidos por um banco para manipular uma visão consolidada do cliente, poderia considerar essa solução um pattern? Sim, mas quantos se beneficiariam de sua utilidade? Isso é muito específico e uma solução que possivelmente pode ser repetida em vez de um pattern.

Em algum local encontra-se o "ponto ideal", em que as idéias principais em todos os níveis envolvidos na criação do sistema podem ser abstraídas e reaplicadas para solucionar vários problemas do mercado. Esses são os patterns mais avançados. Neste artigo, pensamos que o exemplo da especificação de Componente de Negócio mostrado encontra-se nesse ponto ideal para a indústria da saúde. Consideramos isso um aspecto importante dos business patterns que podem ser claramente reconhecidos – alguns não serão de interesse fora da indústria (como este) e outros serão (que representam funções comuns). Especulamos que os comuns que passam pelas indústrias realmente identificam as áreas em que mais se espera a terceirização – mas essa é uma discussão completamente diferente. [0]

[1] Nossa definição de "pattern" segue a definição clássica criada por Christopher Alexander em A Timeless Way of Building, Oxford University Press, 1979. Ao criar patterns utilizamos a notação de estilo de Coplien.

[2] Para obter informações sobre SAM – consulte Enterprise Architecture – Understanding the Bigger Picture, Bob Jarvis, Best Practice Guideline, publicado por National Computing Centre, Reino Unido, maio de 2003 ou http://www.systems-advisers.com

mostrar como usar técnicas padrão para desenvolver descrições de funções, dados e componentes de negócio necessários para um business patterns. Não descrevemos a infra-estrutura necessária e, portanto, também não exploramos as relações com a infra-estrutura. Nosso objetivo é poder dizer "pode ser feito, e você já sabe como" em vez de fornecer um guia detalhado de cada etapa.

Primeiramente mostraremos uma forma para definir as funções, os dados e os componentes de negócio necessários e depois usaremos o PRM3 para mostrar o direcionamento da jornada. O exemplo utilizado baseia-se na indústria de saúde, mas as técnicas são válidas para qualquer indústria. As técnicas mostradas são aquelas realmente usadas em projetos reais. É importante destacar que não estamos dizendo que você deva utilizar estas técnicas. As técnicas por si só não são importantes – são os resultados que importam. Elas devem permitir a criação de patterns mais refinados ou de sistemas de software reais a partir de produtos finais.

Funções de Negócio
Para desenvolver business patterns, nosso primeiro objetivo deve ser descobrir, definir e documentar funções de negócio relevantes. Para isso, trabalharemos com duas tarefas principais:

1. Queremos descobrir o conjunto de funções de nível atômico que descreve o espaço do problema. Esse conjunto é conhecido como "funções primitivas" na modelagem funcional. O mesmo é considerado como "processos elementares" na Reengenharia de Processos de Negócio.
2. Queremos agregar as funções atômicas em grupos funcionais cada vez mais refinados.

[3] Problem Refinement Model – consulte o Artigo 1 do JOURNAL2.

Observe que por "atômico" nos referimos à função que não pode ser dividida de forma significativa – uma vez iniciada, a função deve ser concluída ou anulada. Isso pode ser ainda mais refinado e, portanto, volumoso, do que realmente é necessário para a definição do business patterns. Dessa forma, o que realmente procuramos para um business patterns são as funções de negócio definidas no nível da decomposição em que relações significativas podem ser formadas com as outras esferas, particularmente a esfera de dados. No caso das funções de negócio, consideramos que isso, na prática, esteja aproximadamente no quarto nível da decomposição a partir da raiz da hierarquia. Nesse nível geralmente é possível formular relações CRUD (Criar, Ler, Atualizar e Excluir) com entidades de dados mantidas na esfera Dados.

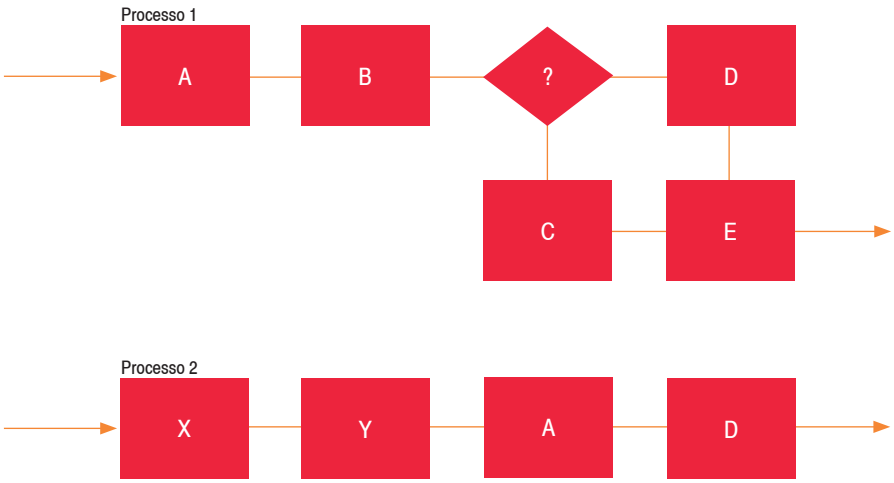
Podemos tratar a definição de funções de negócio de forma bottom-up ou de forma top-down, ou ainda em uma mistura dos dois.

Análise Bottom-Up
Nesta abordagem, o analista trabalha com um conjunto representativo de usuários do domínio de negócio, para analisar sua visão dos processos de negócio. Isso pode ser realizado usando diagramas de caso de uso ou qualquer técnica que possa mostrar as tarefas seqüenciais e paralelas executadas no processo. A coleção de processos é então catalogada e analisada e geralmente se observa que várias das etapas ou tarefas de baixo nível executadas nesses processos são repetidas várias vezes em diferentes processos. As tarefas redundantes são identificadas e um conjunto não redundante de funções primitivas é eliminado.

Isso é exibido na Figura 1 usando uma técnica de fluxograma simples. É possível constatar que a tarefa A é redundante e por isso aparecerá somente uma vez no conjunto primitivo.

Em seguida, após ter derivado um conjunto primitivo não redundante por esta análise, podemos agora agregar de forma repetitiva as funções em uma hierarquia provisória, conforme a mostrada na *Figura 2*.

Do ponto de vista do business patterns, existem problemas com o uso de uma abordagem bottom-up:



1. Isso pode ser um processo muito trabalhoso para grandes áreas comerciais, envolvendo vários usuários (Pode ser atenuado pelo comprometimento em um cenário, mais a abordagem de caso de uso, em vez de uma análise completa do processo).

2. A natureza do processo leva a uma análise da situação "tal como está". Isso é aceitável contanto que seja claro e que os resultados sejam utilizados de acordo.

3. Para certificar-se de que está documentando um pattern, seria necessário repetir ou pelo menos verificar a análise em várias empresas semelhantes. Isso pode apresentar vários problemas práticos.

Os benefícios da abordagem bottom-up são que a análise fundamental está agora concluída e pode prontamente ser aplicada como uma base para uma solução derivada do pattern. O objetivo de um pattern é documentar a prática bem-sucedida, e visto que o exemplo foi bem escolhido, isso naturalmente ocorrerá.

Análise Top-Down

A decomposição funcional na *Figura 2* pode ser derivada de outra forma. Isso pode envolver a criação de hipóteses de níveis superiores na hierarquia e o "preenchimento" dos níveis inferiores até que um grau satisfatório de detalhe seja obtido. Claramente, é necessário verificar se os resultados são precisos e refletem a realidade.

Assim, a análise top-down começa com a descrição do domínio do problema de negócio em sua forma mais abstrata e então é decomposto repetitivamente até que o nível seja alcançado. Isso pode ser feito usando uma abordagem padrão como IDEF04 para a modelagem funcional. De forma alternativa, use as extensões UML para modelagem de processos de negócio.

Na verdade, a abordagem "top-down" nem sequer precisa tratar dos processos de negócio. Existem os prós e os contras disso. Um benefício de usar essa abordagem na definição do pattern é que o trabalho pode ser realizado por meio de consultores da indústria que trabalharam com vários clientes no domínio do problema. Na essência, está sendo realizada uma "mineração do conhecimento" com especialistas e isso pode ser muito mais rápido. Reduz o número de exemplos necessários para trabalhar diretamente, e altera a função do analista como revisor em vez de trabalhador.

A Abordagem Mista

Na prática, é freqüente o caso em que realizamos as abordagens top-down e bottom-up em conjunto, comutando repetitivamente da síntese do processo para a construção de hierarquia para a decomposição ao nível inferior seguinte. Isso possui o benefício de verificar a decomposição hipotética top-down com as funções do mundo real colhidas bottom-up.. Na prática, esse pode ser o método mais rápido e confiável.

Exemplo de Decomposição Funcional

O exemplo seguinte, e os próximos, baseiam-se em situações do mundo real na Assistência Médica e mostram os produtos finais para um business patterns fundamental, usando a abordagem de análise de processo.

Figura 2. Exemplo de um resultado normal de uma análise da função do negócio

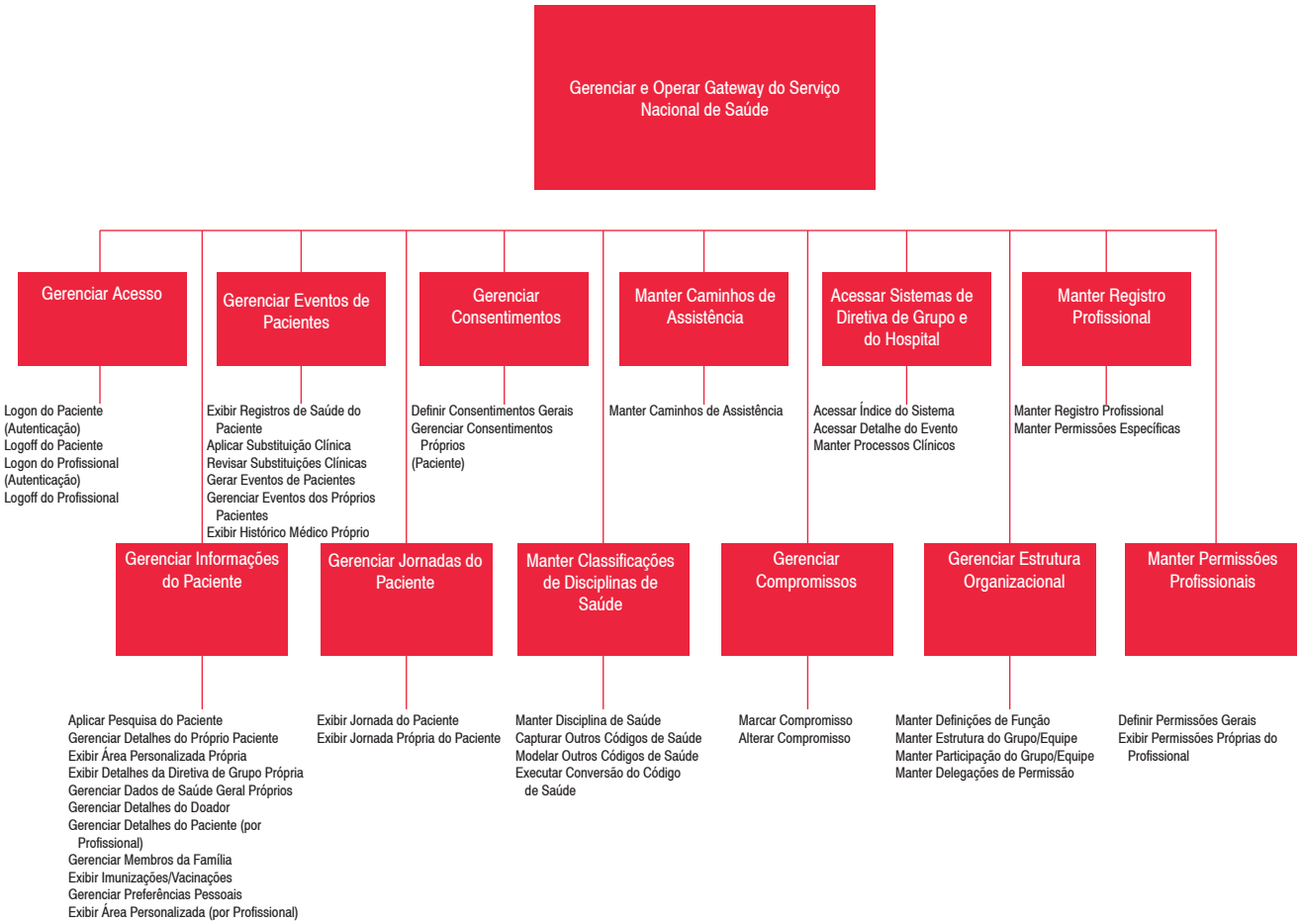
[4] Membro da família IDEF das normas FIPS da NIST. Visite o site http://www.itl.nist.gov/fipspubs/by-num.htm, número 183.

Primeiramente, fizemos uma análise dos requisitos funcionais do domínio do problema. Trabalhando a partir dos cenários fornecidos, como assistência ao câncer de mama, derivamos mais de 40 processos realizados por pacientes, profissionais, administradores de sistema e o "guardião de confidencialidade". Essa última função está encarregada da administração da confidencialidade das informações.

Esses processos foram documentados por meio de casos de uso UML. Esses casos provaram ser altamente repetitivos visto que as subatividades iguais ou semelhantes ocorreram dentro de vários casos de uso. Dessa forma, extraímos e consolidamos essas atividades em uma única lista, descrita a seguir:

- Logon do Paciente (Autenticação)
- Logoff do Paciente (Auditoria)
- Aplicar Pesquisa do Paciente
- Gerenciar Detalhes do Próprio Paciente
- Gerenciar Preferências do Próprio Paciente
- Exibir Área Personalizada
- Exibir Detalhes da Diretiva de Grupo Própria
- Gerenciar Dados de Saúde Geral Próprios
- Gerenciar Detalhes do Doador
- Gerenciar Detalhes do Paciente
- Gerenciar Membros da Família
- Exibir Imunizações/Vacinações
- Gerenciar Preferências Pessoais
- Exibir Registros de Saúde do Paciente
- Exibir Jornada do Paciente
- Exibir Área Personalizada do Paciente
- Aplicar Substituição Clínica
- Revisar Substituições Clínicas
- Gerar Eventos de Pacientes
- Gerenciar Eventos dos Próprios Pacientes
- Construir Jornada do Paciente
- Exibir Jornada Própria do Paciente
- Exibir Histórico Médico Próprio
- Definir Consentimentos Gerais

Figura 3. Exemplo de Assistência Médica – Decomposição Funcional



Gerenciar Consentimentos Próprios
Manter Disciplinas de Saúde
Executar Conversão do Código de Saúde
Capturar Outro Código de Saúde
Modelar Outra Estrutura de Código de Saúde
Manter Caminhos de Assistência
Gerenciar Compromisso Marcado
Alterar Compromisso
Acessar Detalhe do Evento
Acessar Índice do Sistema
Manter Processos Clínicos
Manter Definições de Função
Manter Estrutura do Grupo/Equipe
Manter Participação do Grupo/Equipe
Manter Delegações de Permissão
Manter Registro Profissional
Logon do Profissional (Autenticação)
Logoff do Profissional (Auditoria)
Exibir Permissões Próprias do Profissional
Manter Permissões Específicas
Definir Permissões Gerais

Esses processos podem ser representados em uma hierarquia, conforme mostrado na Figura 3. Ao fazê-lo, derivamos o mais alto nível que resume e contém essas atividades de acordo com a disciplina e a similaridade funcional.

Modelo de Dados
O exemplo da Assistência Médica baseia-se em um modelo de dados abrangente.

Realizamos uma análise dos dados criados e gerenciados dentro do escopo do nosso domínio do problema. Isso revelou 31 entidades principais como Paciente, Profissional de Saúde, Evento do Paciente, Caminho de Assistência e assim por diante. Esses foram definidos. Identificamos chaves primárias candidatas e atributos fundamentais para essas entidades e mapeamos as relações entre elas, incluindo a resolução de qualquer relação muitos-para-muitos. A identificação dessas entidades e de suas relações foi conduzida pela análise funcional realizada em paralelo. Até onde temos conhecimento, todos os requisitos funcionais identificados têm suporte do modelo de dados e vice-versa. As 31 entidades foram alocadas em oito "assuntos de dados", com base na coesão das entidades em termos de força relativa das relações mútuas.

Esses grupos são:
– Pacientes
– Consentimentos do Paciente
– Caminhos de Assistência
– Disciplinas de Saúde
– Processos Clínicos
– Funções, Equipes e Organizações
– Profissionais e Permissões
– Sistemas Locais

Esses assuntos de dados formam a primeira definição dos bancos de dados necessários e de seu conteúdo provisório. O assunto de dado do Paciente é mostrado na Figura 4. Ele leva a forma de um modelo de Relação de Entidade convencional mostrando entidades de dados nomeadas e identificadas pelas chaves primárias. As entidades dentro da área colorida pertencem ao assunto de dados Pacientes. As entidades fora da área colorida pertencem a outros objetos de dados, mas possuem relações significativas com as entidades dentro do assunto de dado Paciente.

A notação de relação de "Pé de Galinha" foi usada, mas uma notação de IDEF1x poderia ter sido usada, se preferível. Todas as relações muitos-para-muitos foram resolvidas. Isso é necessário porque as relações M:M geralmente ocultam outras entidades e relações.

Esta abordagem nos permite formar uma hierarquia superficial para os dados (Raiz > Área do Assunto > Entidade > Atributo) e então formar relações entre os membros de Dados e Função de Negócio. Essa normalidade toma a forma de uma Matriz CRUD mostrando as ações de Funções de Negócio específicas em Entidades de Dados específicas.

Relações de Mapeamento
Depois de ter definido a funcionalidade necessária na forma de uma hierarquia de decomposição funcional e também definido os dados necessários em um modelo de dados de relação de entidade, podemos agora derivar uma primeira tentativa de arquitetura de componente comparando as funções e os dados identificados.

É possível fazer isso formando a matriz, em que linhas representam as funções identificadas e as colunas representam as entidades de dados identificadas. Nas células colocamos um valor:

– "C", que representa a função CRIAR (CREATE) de uma instância dessa entidade de dados.

– "R", que representa a função LER (READ) de uma instância dessa entidade de dados.

– "U", que representa a função ATUALIZAR (UPDATE) de uma instância dessa entidade de dados.

– "D", que representa a função EXCLUIR (DELETE) de uma instância dessa entidade de dados.

O resultado é mostrado na Figura 5.

Matriz Agrupada
Garantimos que cada coluna (entidade de dados) possua pelo menos uma operação de criação e que cada linha (função) possua alguma atividade. Observe que os valores não são absolutamente precisos, particularmente *com relação* às operações de leitura. Não especificamos operações de *exclusão*.

Figura 4. Exemplo de Modelo de Dados – Assunto Dados do Paciente

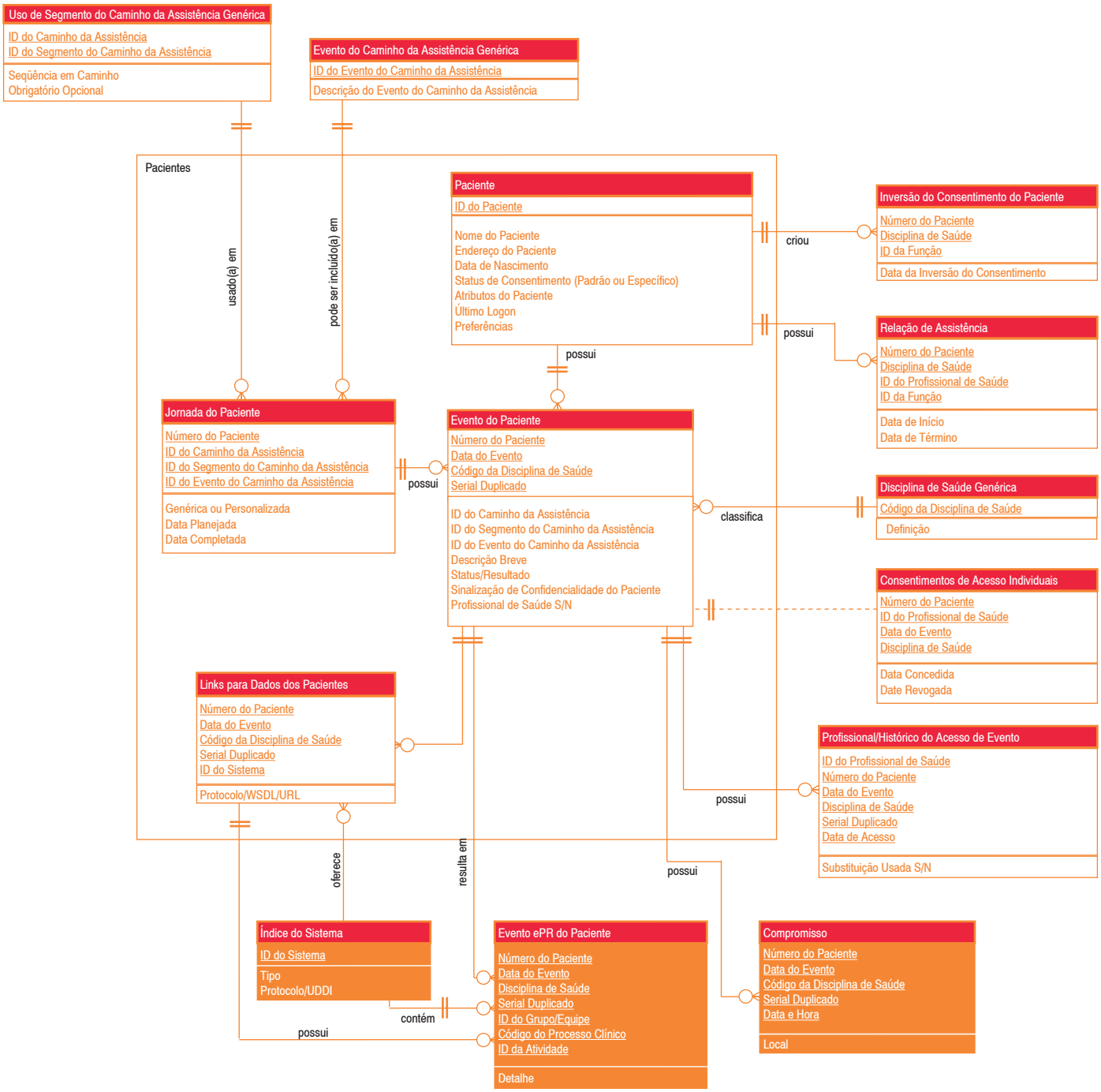


Figura 5. Função v. Matriz "CRUD" de Dados

Arquitetura do Componente de Gateway do Serviço Nacional de Saúde	Compromissos	Relação de Assistência	Caminho Clínico (Local)	Atividade do Caminho Clínico	Tabela de Consentimentos Gerais	Caminho da Assistência Genérica	Evento do Caminho da Assistência Genérica	Segmento do Caminho da Assistência Genérica	Uso de Segmento do Caminho da Assistência Genérica	Participação do Grupo/Equipe	Funções do Grupo/Equipe	Estrutura do Grupo/Equipe	Disciplina de Saúde	Conversão do Código da Disciplina de Saúde	Consentimentos de Acesso Individuais	Grupos e Equipes do Serviço Nacional de Saúde	Profissional do Serviço Nacional de Saúde	Funções dos Profissionais do Serviço Nacional de Saúde	Funções do Serviço Nacional de Saúde	Outro Código de Saúde	Uso de Evento de Segmento do Caminho	Paciente	Inversão do Consentimento do Paciente	Evento do Paciente	Jornada do Paciente	Links para Dados dos Pacientes	Eventos ePR do Paciente	Delegações de Permissão	Permissões Profissionais	Profissional/Histórico do Acesso de Evento	Índice do Sistema
Logon do Paciente (Autenticação)																						C									
Logoff do Paciente																						U									
Aplicar Pesquisa do Paciente		R																				R							R		
Gerenciar Detalhes do Próprio Paciente																						U									
Gerenciar Preferências do Próprio Paciente																						U									
Exibir Área Personalizada																						R									
Exibir Detalhes da Diretiva de Grupo Própria																						U									
Gerenciar Dados de Saúde Geral Próprios																						U									
Gerenciar Detalhes do Doador																						U									
Gerenciar Detalhes do Paciente																						U							R		
Gerenciar Membros da Família																						U							R		
Exibir Imunizações/Vacinações																						R									
Gerenciar Preferências Pessoais																						R									
Exibir Registros de Saúde do Paciente													R		R		R	R				R	R	R		R			R	C	
Exibir Jornada do Paciente						R	R	R	R								R	R			R	R		R	R				R	C	
Exibir Área Personalizada do Paciente																	R					R							R	C	
Aplicar Substituição Clínica																	R					R		R					R	C	
Revisar Substituições Clínicas																	R	R											R	R	
Gerar Eventos de Pacientes	R		R	R									R	R						R		R		C		C	R		R		R
Gerenciar Eventos dos Próprios Pacientes																						R		U							
Construir Jornada do Paciente						R	R	R	R												R	R		U	C				R		
Exibir Jornada Própria do Paciente																						R		R	R						
Exibir Histórico Médico Próprio																						R		R							
Definir Consentimentos Gerais					C								R		U								U								
Gerenciar Consentimentos Próprios					R								R		C				R			R	C								
Manter Disciplinas de Saúde													C	C						C											
Executar Conversão do Código de Saúde													R	C						R											
Capturar Outro Código de Saúde																				C											
Modelar Outra Estrutura de Código de Saúde																				C											
Manter Caminhos de Assistência						C	C	C	C												C										
Gerenciar Compromisso Marcado	C					R			R													R		R	R				R		
Alterar Compromisso	U																					R		R							
Acessar Detalhe do Evento				R																							R				
Acessar Índice do Sistema																															R
Manter Processos Clínicos			C	C																											
Manter Definições de Função											R								C												
Manter Estrutura do Grupo/Equipe										R	C	C				C			R									R			
Manter Participação do Grupo/Equipe										C	R	R				R	R	R										R			
Manter Delegações de Permissão										R	R	R																C			
Manter Registro Profissional																	C	C	R												
Logon do Profissional (Autenticação)																	U												R		
Logoff do Profissional																	U														
Exibir Permissões Próprias do Profissional										R			R		R		R	R											R		
Manter Permissões Específicas		C								R	R		R			R			R			R						R	C		
Definir Permissões Gerais													R																	C		

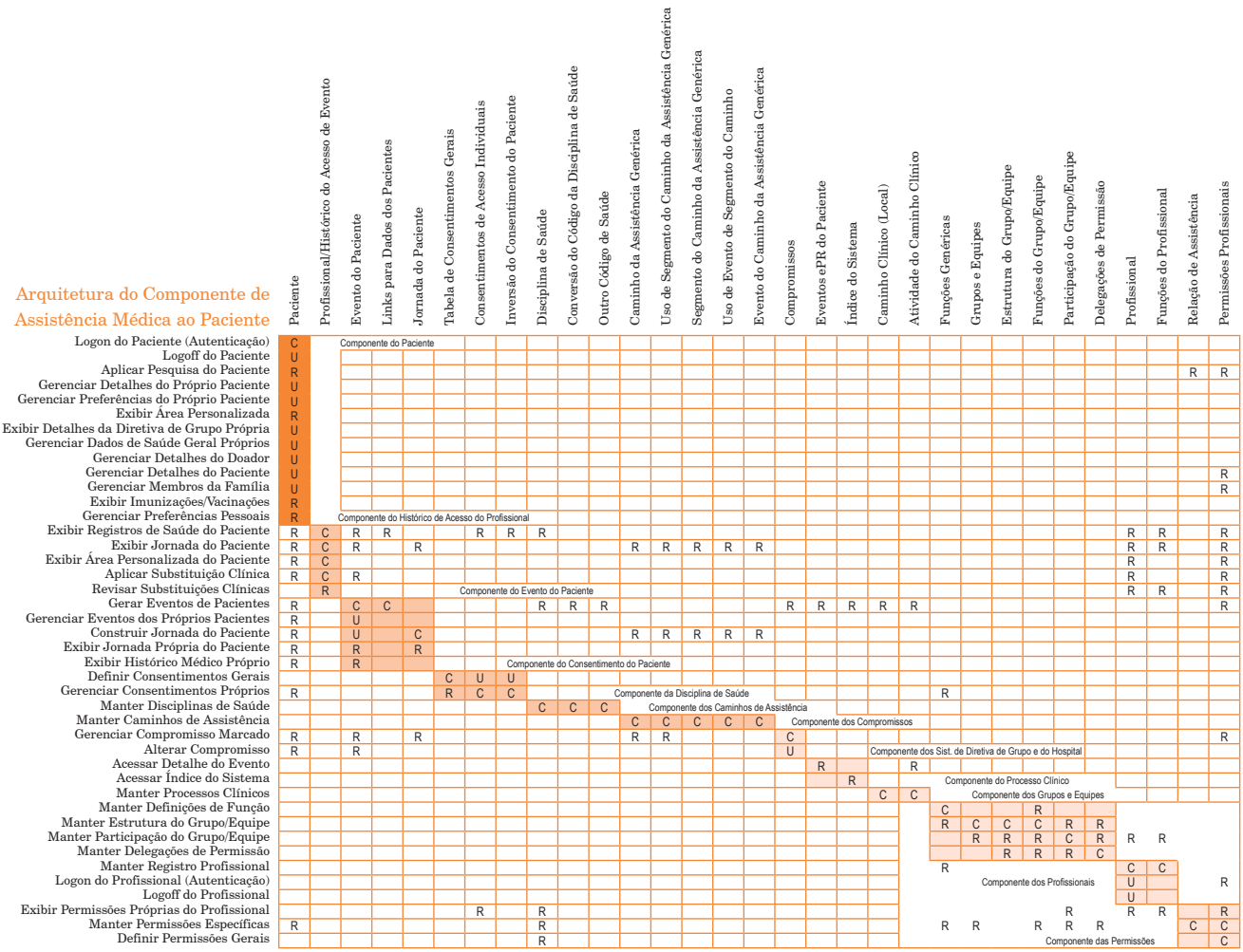
Agora analisaremos a matriz usando a análise de afinidade e a técnica de agrupamento. O objetivo é deduzir grupos de funções e entidades que compartilham operações de criação e atualização. Estamos explorando as noções de "agrupamento flexível" e "coesão estreita" usadas na decomposição funcional com relações interentidades (algumas das quais são vitais e outras meramente transientes) no modelo de dados.

Ajustamos o modelo para unir funções e entidades com forte afinidade. A descrição do algoritmo detalhado usado (o método "Noroeste") está além do escopo deste artigo; no entanto, o resultado é o desenvolvimento de grupos mutuamente exclusivos de funções e entidades formadas em torno das ações Criar e Atualizar. Esses grupos são nossos componentes de negócio candidatos, conforme mostrado na *Figura 6*.

Derivação do Componente de Negócio

Primeiramente, devemos esclarecer o que queremos dizer por componente de negócio. Uma função de negócio cria, lê, atualiza e exclui dados. O agrupamento de todas essas funções que criam e atualizam as mesmas entidades de dados por meio de técnicas como o agrupamento comutativo, define "blocos de construção" não-redundantes – componentes de negócio – que podem ser usados para construir patterns, sistemas ou aplicações que, por sua vez, dão suporte a processos de negócio específicos. A esfera Componentes de Negócio é um exemplo de uma

Figura 6. Matriz Agrupada



"esfera derivada" – uma que é deduzida das relações entre outras duas esferas. Essa é uma técnica potente que explora valores ocultos em uma arquitetura empresarial. Ao encapsular a funcionalidade e os dados em componentes, o reuso e a substituição de software tornam-se práticos. Além disso, os componentes oferecem "serviços" que podem ser "orquestrados" em conjunto com outros serviços oferecidos por outros componentes para criar uma aplicação.

Os componentes de negócio que resultam de nossa análise de cluster incluem interfaces de serviços, componentes de entidades de negócio, componentes de acesso de dados e, talvez, agentes de serviço. Esses artefatos alinham-se com a Arquitetura de Aplicações .Net. O componente de negócio não contém elementos "ágeis" como Componentes de Interface do Usuário e componentes do processo da interface do usuário, fluxos de negócio ou elementos como segurança, gerenciamento operacional ou comunicação.

Pensamos que essa definição grosseira é muito útil. Ela fornece uma parte estável da arquitetura de solução geral que é então arredondada com os elementos ágeis – como Interface do Usuário, Processos da Interface do Usuário e Fluxos de Trabalho de Negócio. *Dessa forma podemos fornecer uma solução ágil baseada na fundação de um pattern estável*.

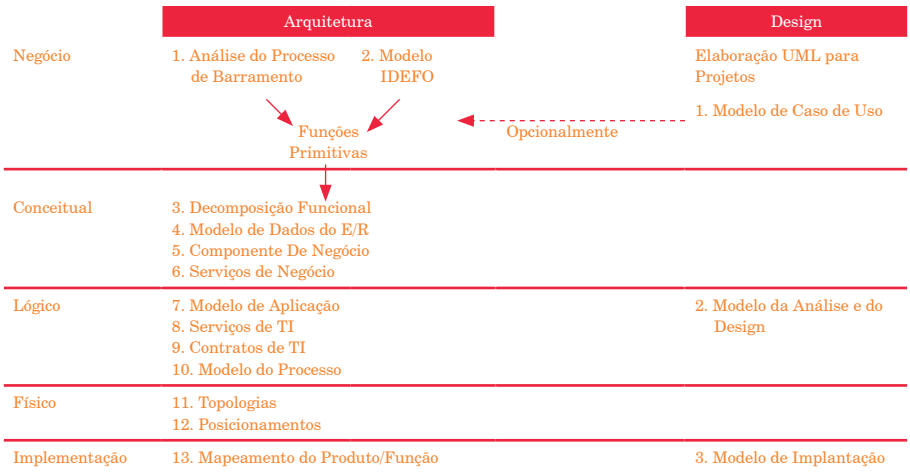
Esses componentes de negócio grosseiros e seus serviços são descobertos por meio de uma análise

[5] Application Architecture for .Net: Designing Applications and Services – Patterns and Practices Guide (Arquitetura de Aplicações para .Net: Criação de Aplicações e Serviços – Guia de Padrões e Práticas) – Microsoft Corporation 2002

Figura 7. Amostra de Componente de Negócio

Especificação de Componente de Negócio		
Componente do Paciente		
Descrição		
O componente Paciente oferece suporte ao armazenamento e ao acesso a dados relacionados a algum paciente. Funções são fornecidas para introduzir, validar, manter, armazenar e retirar dados de pacientes permanentes, como nome e endereço, detalhes pessoais e dados limitados relacionados à medicina, caso estejam disponíveis.		
Funções com suporte (Lógica de Negócio)	Dados mantidos (Entidades de Negócio)	
EXEMPLOS: Logon do Paciente (Autenticação) Logoff do Paciente (Auditoria) Aplicar Pesquisa do Paciente Gerenciar Detalhes do Próprio Paciente Exibir Detalhes da Diretiva de Grupo Própria Gerenciar Dados de Saúde Geral Próprios Gerenciar Detalhes do Doador Gerenciar Dados do Paciente (por Profissional) Gerenciar Membros da Família Exibir Imunizações/Vacinações Gerenciar Preferências Pessoais Etc.	EXEMPLOS: Número do Paciente Nome do Paciente Endereço do Paciente Data de Nascimento Atributos do Paciente: Telefone Profissão Sexo Necessidades Especiais Estado Civil Histórico do Peso Etc…	PK AK AK AK Atts
Serviços e Recursos fornecidos:		
EXEMPLOS: • Registros de logon e logoff do paciente • Pesquisa de paciente por Número do Paciente ou Nome/Endereço/Data de Nascimento etc. • Manutenção dos Detalhes do Paciente e das Preferências por Paciente • Fornecimento de atribuições de diretivas de grupo para cada paciente • Fornecimento de atributos médicos do paciente (grupo sangüíneo, etc.) – fornecidos pela diretiva de grupo • Fornecimento de informações sobre imunização/vacina – fornecidas pela diretiva de grupo • Etc.		

Figura 8. PRM com visões de Arquitetura e Design



de afinidade e agrupamento executada nas relações entre as funções de negócio e os dados. Depois que essas esferas são definidas, a matriz CRUD é criada para determinar a relação componente-dados.

No SAM, componentes de negócio também formam uma hierarquia. Dessa forma, o componente de negócio pode decompor um nível para especificar os serviços, a entidade de negócio e os subcomponentes de acesso de dados. Nesse nível, os componentes de negócio também especificam os serviços de negócio que pretendem expor. É útil explicitar isso, de forma que as decisões possam ser tomadas com base na opção escolhida de apresentar esses serviços como internos ou como serviços web.

Componentes e Serviços

A lista de componentes é elaborada da seguinte forma:

– Componente do Paciente
– Componente do Histórico de Acesso Profissional
– Componente dos Eventos do Paciente
– Componente dos Consentimentos do Paciente
– Componente da Disciplina de Saúde
– Componente dos Caminhos de Assistência
– Componente dos Compromissos
– Componente do Acesso a Sistemas de Diretiva de Grupo e do Hospital
– Componente dos Processos Clínicos
– Componente dos Grupos e Equipes
– Componente dos Profissionais
– Componente das Permissões

Acreditamos que esse conjunto de componentes representa a essência da definição de Business Pattern.

Um desses componentes, o componente Paciente, é mostrado na Figura 7. Ele indica a funcionalidade, os dados gerenciados e os serviços e recursos oferecidos pelo componente.

Figura 9. Ponto ideal dos Business Patterns no PRM

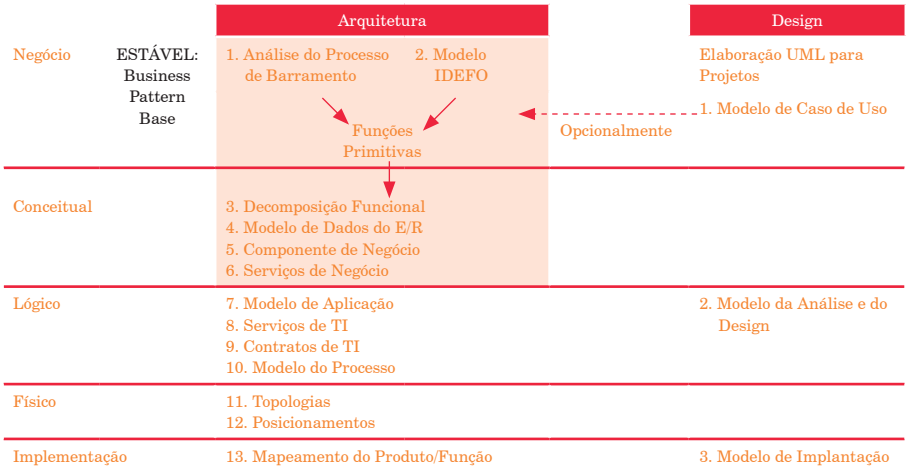
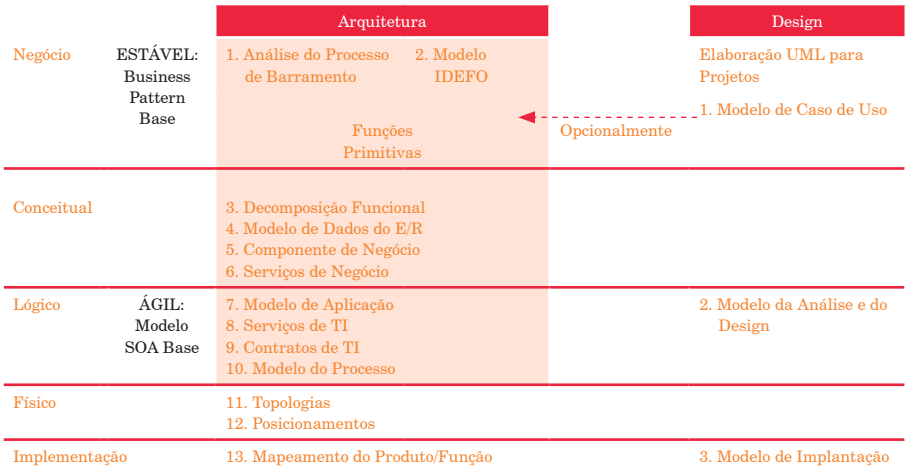


Figura 10. Adição de uma arquitetura de TI aos Business Patterns



Como os Business Patterns são úteis para a Engenharia de Software

Agora chegamos à segunda parte deste artigo, onde falamos sobre o direcionamento de como usar as descrições de Business Patterns mencionadas anteriormente para projetar outros tipos de patterns ou sistemas de software reais.

Ilustramos isso usando as cinco camadas do PRM mostrado na Figura 8. Será possível observar que o PRM distingue a visão do arquiteto (uma visão de nível superior, por exemplo, orientação de uma programação de projetos) e a visão do designer (por exemplo, o design de um projeto ou de um subconjunto disso). A Figura 8 pretende indicar as diferentes técnicas e notações apropriadas a essas visões diferentes, apenas para mostrar brevemente como é feito o aperfeiçoamento.

É importante notar que o PRM apenas identifica as necessidades dos elementos para ir diretamente a uma implementação – não faz suposições ou julgamentos sobre como realizar isso da melhor forma! É possível utilizar qualquer abordagem preferida para preencher o conjunto de produtos finais na ordem que desejar (após a definição dos Business Patterns). Você escolhe a maneira mais apropriada para a organização na qual está trabalhando.

Ao considerar Business Patterns, precisamos reconhecer a diferença essencial entre arquitetura e design no contexto da barra lateral sobre *o nível correto de abstração*.

Onde os Business Patterns Encaixam-se

O melhor uso para os business patterns é exibido na Figura 9. Aqui eles definem os elementos estáveis de um negócio, as funções de negócio, os dados e os componentes de negócio que se encontram no escopo para a iniciativa. Dessa forma, podem ser usados para orientar grandes programações de trabalho, que levam à maior consistência entre projetos com menos esforço geral.

Business Patterns e Arquitetura Orientada a Serviços

Para aqueles que pretendem fornecer outros padrões de TI ou mais orientações relacionadas ao fornecimento de uma solução de TI para o problema de negócio, os

[6] Visite o site http://www.microsoft.com/resources/practices/ ou o Amazon.
[7] http://www.martinfowler.com/books.html#eaa
[8] Design Patterns Elements of Reusable Object-Oriented Software por Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. http://hillside.net/patterns/DPBook/DPBook.html

Figura 11. Adição de elementos de tecnologia da Microsoft

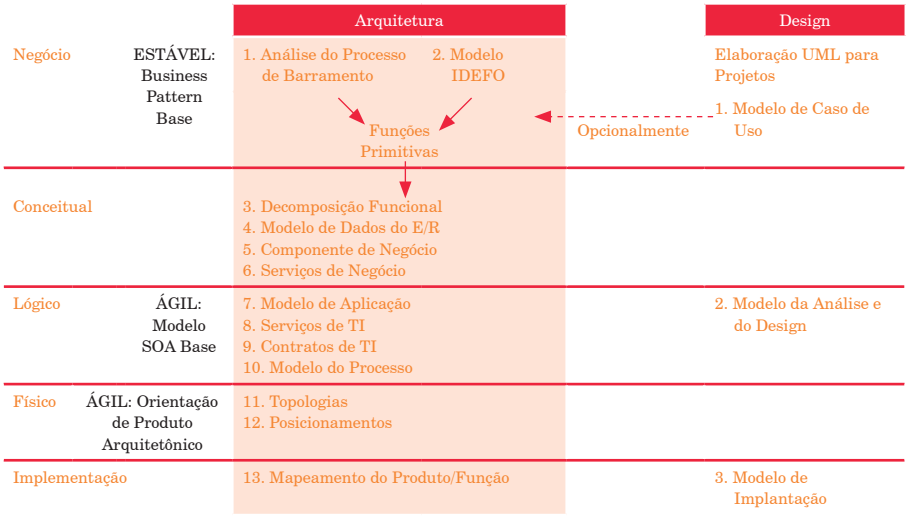
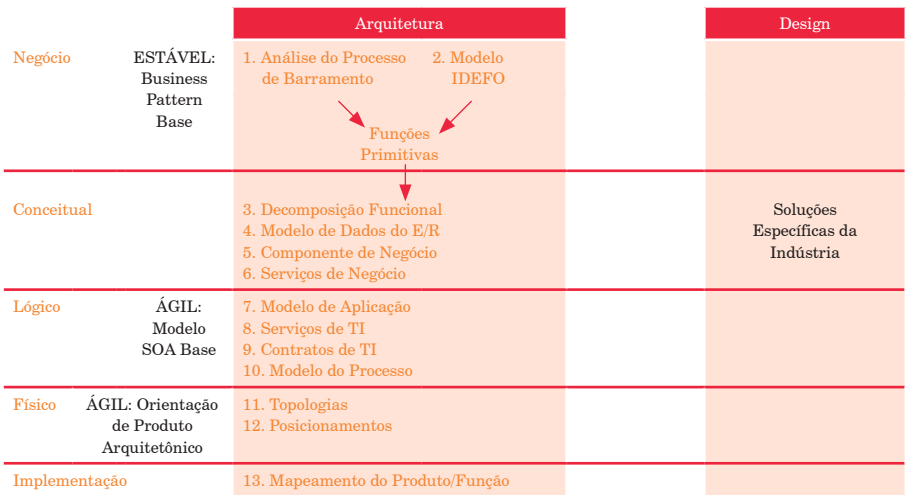


Figura 12. Refinamento para o design da solução



Business Patterns podem ser mais aperfeiçoados com os elementos de uma Arquitetura Orientada a Serviços (SOA não é necessária, mas é uma opção excelente; o que é necessário é uma elaboração arquitetônica para transformá-la em uma solução de TI).

Para fazer isso precisamos adicionar os tópicos que mencionamos estarem de acordo com oportunidades para aumentar a agilidade de negócio, conforme exibido na *Figura 10*.

Os conceitos que estamos descrevendo aqui e na parte seguinte podem ainda ser descritos na forma de patterns. A adição desses dois conjuntos de elementos muda um business pattern para um business solution pattern. Alternativamente, podemos criar uma arquitetura de solução de negócio específica para o business pattern. Nós revisaremos isso em alguns momentos.

Observe que neste estágio todo o trabalho ainda é independente de produtos tecnológicos.

Business Patterns, SOA e detalhes de produtos (Microsoft)
Finalmente podemos elaborar ainda Finalmente podemos elaborar ainda mais com as recomendações práticas e de produto para implementar com sucesso a solução na tecnologia Microsoft, conforme visto na Figura 11. Neste ponto, apresentamos os Business Patterns, uma arquitetura de aplicações orientada a serviços e um conjunto de patterns Microsoft para implementação. Exemplos de patterns tecnológicos da Microsoft podem ser encontrados no site http://www.microsoft.com/resources/practices/

Robert Jarvis,
Diretor, SA Ltd
v-rjarvi@microsoft.com
Robert Jarvis é Diretor da Systems Advisers Limited, uma empresa de consultoria no Reino Unido especializada no desenvolvimento de Arquiteturas Estratégicas de Sistemas para as principais empresas internacionais.
Ele também é Consultor de Arquitetura Associado com a Microsoft Ltd. Bob tem mais de 30 anos de experiência como Arquiteto e Consultor de Sistemas Internacionais orientando organizações empresariais e governamentais no Reino Unido, na Europa Continental e nas Américas. É especializado em Arquitetura Empresarial, trabalhando especificamente na intersecção negócios/tecnologia. Ele é autor do livro "Enterprise Architecture – Understanding the Bigger Picture", um guia de práticas recomendadas publicado pelo National Computing Centre no Reino Unido em 2003.

Neste estágio poderíamos ter completado a oferta de um conjunto completo de padrões ligados, ou de uma solução de TI, para solucionar problemas de negócio comuns no nível da arquitetura. Mas e se você realmente quiser criar um sistema? Provavelmente precisará de mais detalhes, que é o que a visão do design oferece.

Soluções de Business Patterns e Soluções da Indústria
As propostas anteriores fornecem uma arquitetura de orientação que pode ser usada para estabelecer o escopo, o custeio e o controle de projetos de TI que a implementam. No entanto, não são detalhadas o suficiente para a implementação da solução. Se quisermos fornecer uma solução, precisamos elaborar mais o nível do design, que é orientado pela arquitetura fornecida. É provável que a solução seja implementada em vários projetos e, nesse caso, a arquitetura é o que mantém todos os projetos no caminho correto para a consistência entre eles.

A *Figura 12* mostra esse aperfeiçoamento. Na figura mostramos como o UML é uma forma comum para conduzir a fase de design. Novamente, queremos ser claros que isso é uma ilustração e que não há nada que diga que isso deva ser feito dessa forma. Tudo o que dizemos é que essa é uma forma comum de obter o aperfeiçoamento da arquitetura de um design para implementação.

Enquanto estamos realizando esse aperfeiçoamento, podemos novamente criar patterns – designs de TI neste momento. Na verdade, estamos agora chegando (finalmente) ao território que a maioria das literaturas sobre patterns de software da atualidade descreve! Exemplos incluem os Business Solution Patterns da Microsoft Usando o .NET e os Padrões de Martin Fowler da Arquitetura de Aplicações Empresariais.

Ou, em vez de padrões, podemos criar um design real de solução.

Este é o final do direcionamento do artigo. Claramente, a última etapa seria implementar o design, e é por isso que padrões como Gangue dos Quatro8 e o conjunto de Arquitetura de Software Orientada a Padrões são muito relevantes, bem como os patterns Microsoft e de Martin Fowler, mencionados anteriormente.

Resumo – Estrutura e abordagem para Business Patterns e Soluções da Indústria
O que descrevemos neste artigo é uma estrutura e uma abordagem para criar e usar Business Patterns por meio de uma arquitetura de orientação que pode ser usada para fazer o escopo, custear e controlar projetos de TI para implementação. Para fornecer uma solução, é necessária mais elaboração no nível do design, e isso é orientado pela arquitetura fornecida.

Aviso de Isenção de Responsabilidade: As opiniões expressas neste artigo são as dos autores. Elas não são necessariamente endossadas por suas empresas e não há implicação que qualquer uma dessas idéias ou conceitos sejam proporcionados como ofertas ou produtos por essas empresas.

Philip Teale,
Consultor de Estratégias de Parcerias,
Microsoft Reino Unido
pteale@microsoft.com

Philip Teale é Consultor de Estratégia de Parcerias e trabalha para o Enterprise & Partner Group na Microsoft Reino Unido. Anteriormente, ele trabalhou para o Grupo de Arquitetura Prescritiva da Microsoft em Redmond e, antes disso, para os Serviços de Consultoria da Microsoft. Tem 29 anos de experiência em TI Empresarial, dos quais quatro anos foram na Microsoft e 16 na IBM, em ambos os locais desempenhando as funções de desenvolvimento de software e campo. Sua experiência internacional inclui nove anos de trabalho nos Estados Unidos, três anos no Canadá e 17 no Reino Unido. A experiência anterior de Phil inclui arquitetura, design e criação de grandes sistemas de negócio complexos distribuídos. Sua mais recente contribuição para a indústria por meio do papel de líder foi conduzir a Microsoft na criação de patterns para o desenvolvimento de sistemas empresarias. Philip é Membro da RSA.

Abordagem Estratégica dos Métodos de Transferência de Dados

Por E G Nadhan e Jay-Louise Weldon, EDS

Introdução
Atualmente, os negócios são conduzidos pelo acesso às informações corretas no momento correto. A informação é o salva-vidas da empresa. No entanto, o acesso oportuno às informações corretas é dificultado pela variedade e complexidade de aplicações de negócio e pelos volumes crescentes de dados mantidos. Os dados precisam ser compartilhados para que os processos de negócio sejam eficazes em toda a empresa. As organizações possuem uma variedade de formas de compartilhamento de dados. Um conjunto completamente integrado de aplicações ou o acesso a bancos de dados comuns é o ideal. Entretanto, se essas alternativas não forem práticas, os dados devem ser movidos de uma aplicação ou banco de dados para outro e o designer deve escolher entre a variedade de alternativas existentes para a transferência de dados. Como resultado disso, o compartilhamento de dados apresenta-se como um desafio significativo na ausência de uma estratégia de transferência de dados estabelecida para a empresa.

A maioria das empresas adquiriu ou criou aplicações que oferecem suporte à execução de processos de negócio específicos às unidades de negócio autônomas dentro da empresa. Embora essas aplicações sirvam para a necessidade específica das unidades de negócio, continua a existir uma necessidade de se compartilhar dados coletados ou mantidos por essas aplicações com o restante da empresa. Nos casos em que as aplicações servem como os Sistemas de Registro, o volume de dados a ser compartilhado é relativamente alto. Além disso, as empresas acumularam grandes volumes de dados durante as últimas décadas, visto que os custos de armazenamento diminuíram e que a quantidade de atividade rastreada no ambiente do comércio eletrônico cresceu além do mundo centrado em mainframes. As organizações de TI modernas enfrentam o desafio de armazenar, gerenciar e facilitar a troca de dados a volumes sem precedentes. Embora os sistemas de gerenciamento de bancos de dados tenham evoluído para atender ao armazenamento e ao gerenciamento de terabytes de dados, o problema de se trocar eficazmente altos volumes de dados entre as aplicações empresariais ainda permanece. Uma estratégia de transferência de dados sólida em toda a empresa é necessária para guiar os profissionais de TI e também para permitir uma representação consistente das principais entidades de negócio entre as aplicações empresariais.

Histórico
O mundo de mainframe nos anos 70 consistia de aplicações monolíticas orientadas por cartões perfurados, muitas das quais continuam a ser os sistemas de registros em organizações da atualidade. A chegada dos Computadores Pessoais nos anos 80 promoveu o mundo cliente-servidor do início dos anos 90, quando o PC evoluiu para estações de trabalho robustas. O poder de processamento continuou a crescer exponencialmente, resultando na introdução de servidores departamentais que foram empregados pelas principais unidades de negócio dentro das organizações. A tecnologia cliente-servidor ofereceu a essas unidades de negócio autônomas o poder para armazenar e usar dados em seus próprios mundos. Tal autonomia gerou vários repositórios que eram eficientes em armazenar pacotes isolados de dados dentro da empresa. A computação distribuída em n camadas no final dos anos 90 resultou na criação de camadas adicionais que armazenavam dados específicos dessas unidades de negócio.

Embora os processos de negócio de departamentos não sofram impactos pela proliferação de dados entre vários repositórios, existe uma necessidade de aproveitar os dados no nível empresarial – bem como no nível das unidades de negócio. Por exemplo, as organizações precisam ter uma visão de nível empresarial do cliente e servir esses clientes como uma única entidade lógica. No mundo atual da interação on-line e em tempo real com clientes, a resposta de sistemas ponto-a-ponto também se tornou um fator crítico de sucesso. Os requisitos fundamentais para fornecer serviços básicos ao cliente não mudaram com o passar dos anos. No entanto, a manutenção e a recuperação convenientes de dados para fornecer tais serviços tornaram-se um processo muito mais complexo.

Apesar dessa complexidade, as empresas atuais precisam ter acesso a todos esses pacotes de dados, bem como aos sistemas originais de registro. Tal acesso é obtido pela criação de mecanismos de conexão a vários sistemas ou pela transferência de dados entre sistemas em intervalos periódicos. As ferramentas EAI (Enterprise Application Integration) podem ser aplicadas para mover transações e mensagens de uma aplicação para outra. As ferramentas ETL (Extract Transformation and Load) executam a mesma tarefa, mas geralmente movem os dados em lotes.

Este artigo descreve as opções disponíveis para solucionar esse problema de compartilhamento de dados. Embora as opções não sejam mutuamente exclusivas, elas representam princípios de design e execução diferentes logicamente.

Público-Alvo
Equipes de TI que enfrentam os desafios do compartilhamento de dados entre várias aplicações dentro da empresa podem beneficiar-se do conteúdo deste artigo. Tais equipes incluem Arquitetos de TI Empresariais, Arquitetos de Dados,

"Uma estratégia de transferência de dados sólida em toda a empresa é necessária para guiar os profissionais de TI e também para permitir uma representação consistente das principais entidades de negócio entre as aplicações empresariais."

Arquitetos de Integração, bem como Especialistas no Assunto das principais aplicações empresariais. Gerentes de Processos e Funcionais dentro das empresas que trabalham perto dos arquitetos de TI desenvolverão uma apreciação pelas complexidades do compartilhamento de dados conduzido pelas alterações em processos de negócio.

Definição do Problema
Geralmente as aplicações precisam tornar seus dados acessíveis a outras aplicações e bancos de dados por vários motivos. Os dados podem precisar ser movidos de uma plataforma para outra ou de um local geográfico para outro. Os dados podem precisar ser movidos para torná-los acessíveis a outras aplicações que os necessitam sem causar impactos no desempenho do sistema de origem. Alterações nos dados podem precisar ser movidas para manter os dois sistemas sincronizados. Freqüentemente, empresas criam um repositório compartilhado, chamado ODS (Operational Data Store), para coletar dados de sistemas de origem e torná-los disponíveis a outros sistemas e bancos de dados. Os dados devem, então, ser movidos da aplicação de origem para o ODS.

Existem várias formas de realizar a transferência de dados e vários fatores a serem considerados ao escolher a alternativa que melhor se encaixa à situação. A eficiência é fundamental quando o volume de dados é grande. A transferência de dados em lote pode não ser uma alternativa viável devido a limitações de tempo. Ao mesmo tempo, a identificação dos dados alterados pode ser um desafio.

O exemplo a seguir representa uma situação real em que isso se manifesta.

Cenário Exemplo
Um site voltado a clientes oferece aos assinantes um serviço de registro on-line. O processo envolvido nessa atividade irá capturar uma variedade de elementos de dados relevantes sobre o assinante. Os dados capturados serão armazenados imediatamente em um sistema de Vendas (por exemplo, um Sistema de Gerenciamento de Pedidos). Neste exemplo, o sistema de Vendas seria considerado o Sistema de Registro para esses elementos de dados. Os dados do assinante são, sem dúvidas, fundamentais para o departamento de vendas. Ao mesmo tempo, também são importantes para várias outras unidades de negócio. Por exemplo, o departamento de cobranças precisará certificar-se de que as transações financeiras com o assinante são executadas. Além disso, o departamento de marketing poderá precisar desses dados para ajudar a criar campanhas para efetuar vendas cruzadas e ampliadas de produtos e serviços ao assinante. Por isso, é essencial que os dados do assinante sejam inseridos o quanto antes em um ODS, de onde poderão estar disponíveis para as aplicações que precisarem deles.

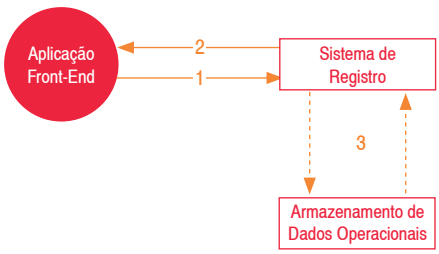
Do ponto de vista dos sistemas, a ***Figura 1*** representa a situação recém-descrita. A figura ilustra uma aplicação front-end armazenando dados em seu Sistema de Registro proprietário e recebendo uma confirmação de atualização bem-sucedida. Esse Sistema de Registro é constantemente preenchido com grandes volumes de dados que precisam ser transferidos para um ODS de forma que possam ser compartilhados com o restante da empresa. As seções subseqüentes ilustram as diferentes formas de efetuar essa transferência.

A ***Figura 1*** ilustra as seguintes etapas:
1. A aplicação front-end atualiza o Sistema de Registro.
2. O Sistema de Registro confirma a atualização bem-sucedida.
3. Transferência de dados para o ODS.

Dependendo do contexto do domínio do problema específico de uma determinada empresa, existem várias abordagens para efetuar a transferência de dados ao ODS nesta situação. As várias abordagens envolvidas são descritas nas próximas seções.

As abordagens apresentadas baseiam-se nas seguintes suposições:
1. Para simplificar, supomos que existe apenas um Sistema de Registro dentro da Empresa para qualquer elemento de dados. A propagação dos dados em vários Sistemas de Registro pode ser realizada usando-se uma ou mais dessas opções.
2. Uma atualização do Sistema de Registro pode indicar uma criação, modificação ou até mesmo uma exclusão de algum registro lógico.
3. A etapa de confirmação é a mensagem final à Aplicação Front-End indicando que todas as etapas intermediárias envolvidas na propagação de dados ao Sistema de Registro, bem como ao ODS, foram concluídas com sucesso. Etapas adicionais de confirmação entre os pares de nós selecionados podem ser necessárias dependendo do contexto de implementação para as situações de negócio específicas.
4. Os metadados, embora não sejam discutidos diretamente neste artigo, são considerações essenciais para a transferência de dados1. Supõe-

Figura 1. Situação de Amostra



se que todas as opções discutidas implicam a captura, manipulação e transferência de metadados. No entanto, a discussão deste artigo limita-se ao fluxo lógico de dados entre os diferentes nós dentro de um processo ponto-a-ponto.

Revisão do Processo de Negócio
Existem várias opções disponíveis para a engenharia da transferência de dados dentro do Cenário Exemplo definido na *Figura 1*.

No entanto, antes de exercitar qualquer opção, é prudente voltar atrás, revisar e validar a necessidade de negócio da transferência de dados. A transferência real de dados entre sistemas pode ser uma manifestação física de problemas diferentes em um nível de processo de negócio lógico. Uma revisão dos processos ponto-a-ponto pode expor oportunidades para modernizar o fluxo do processo de negócio, resultando na racionalização das aplicações constituintes. Tal racionalização pode atenuar e, em alguns casos, eliminar a necessidade de tal transferência de dados. Algumas perguntas simples incluiriam:
– *Por que os dados precisam ser transferidos?*
– *Por que os dados não podem permanecer em um único sistema?*

Uma resposta viável a essas perguntas pode eliminar a necessidade de tal transferência de dados.

Se ainda houver uma clara necessidade dessa transferência de dados, mesmo após uma revisão dos processos de negócio ponto-a-ponto, existem várias opções disponíveis que estão em ampla conformidade com uma ou mais destas abordagens:
– Tecnologias EAI
– Tecnologias ETL
– Combinações

As opções restantes exploram essas diferentes possibilidades.

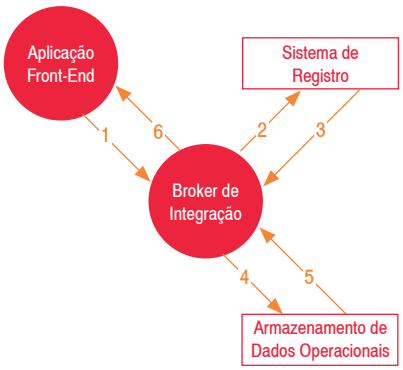
Opções de Transferência de Dados
Esta seção descreve as opções arquitetônicas disponíveis para o compartilhamento de dados entre aplicações distintas. A discussão aqui é independente de soluções de negócio; em vez disso, concentra-se nos gêneros tecnológicos disponíveis no mercado atual.

As opções discutidas nesta seção são:
– Opção 1: Transferência EAI em Tempo Real
– Opção 2: Propagação EAI de Registros Incrementais
– Opção 3: Transferência Incremental em Lotes (Captura de Dados Alterados)
– Opção 4: Replicação Nativa
– Opção 5: Atualização em Lote Usando a Transferência de Arquivo em Lotes
– Opção 6: Transferência ETL/ELT
– Opção 7: Integração de Informações Empresariais

Opção 1: Transferência EAI em Tempo Real
A ***Figura 2*** ilustra a maneira como o Broker de Integração EAI2 pode facilitar essa transferência.

Esta opção é orientada por aplicações e é a mais apropriada para transferências de dados em que as atualizações ao Sistema de Registro e ao ODS fazem parte da mesma transação. Um Broker de Integração recebe a transação iniciada pela Aplicação Front-End e depois assume a responsabilidade pela propagação dos dados ao Sistema de Registro e ao ODS. As etapas são executadas nesta seqüência:
1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Broker de Integração recebe esta atualização da Aplicação Front-End e envia-a ao Sistema de Registro.
3. O Sistema de Registro confirma a atualização.
4. O Broker de Integração inicia imediatamente a atualização correspondente ao ODS, efetuando, dessa forma, a transferência em tempo real desses dados.
5. O ODS confirma o recebimento dessa atualização.

Figura 2. Transferência EAI em Tempo Real



[1] O compartilhamento de dados eficaz requer uma compreensão comum do significado e da estrutura dos dados ao fornecedor e ao receptor. Os metadados – dados sobre dados – são os veículos para obter tão compreensão. Quando algum dado é compartilhado ou fisicamente transferido entre partes, os metadados também devem ser trocados. É de responsabilidade do designer garantir que os metadados corretos sejam capturados e transferidos em todas as situações de transferências de dados.

[2] Um broker de integração é um componente que roteia as mensagens trocadas entre aplicações. Facilita a transferência condicional de mensagens entre aplicações baseadas em regras pré-definidas orientadas pela lógica de negócio e pelos requisitos de sincronização de dados.

6. O broker de integração envia uma confirmação dessas atualizações à Aplicação Front-End.

Situação de Uso – Instituição Financeira
Uma aplicação CRM front-end captura dados sobre chamadas de clientes potenciais ao Centro de Contatos. A aplicação CRM propaga os dados do cliente a um ODS que contém os dados básicos do cliente para referência em toda a empresa. Esses dados precisam ser propagados ao ODS imediatamente de forma que os dados mais atuais estejam disponíveis a todas as outras aplicações destinadas ao cliente, como caixas eletrônicos, centros financeiros (agências) e acesso bancário on-line.

Opção 2: Propagação EAI de Registros Incrementais
Esta opção é orientada por a aplicações e apropriada para dados de menor prioridade. A Aplicação Front-End atualiza o Sistema de Registro, após o qual os dados são propagados ao ODS por meio do CorretorBroker de Integração. Esta é uma característica das situações em que existe um portal estreitamente agrupado ao sistema CRM ou ERP. Existem dois mecanismos diferentes para efetuar a transferência de dados ao ODS:

– *Opção 2a: Envio ao Broker de Integração: O Sistema de registro inicia a notificação do recebimento desses dados ao Broker de Integração. O "envio" freqüentemente é acionado pelo requisito programado, por exemplo, uma atualização diária.*
– *Opção 2b: Recebimento do Broker de Integração: O Broker de Integração consulta continuamente o Sistema de Registro para recebimento desses dados. O "recebimento" freqüentemente é acionado por algum evento de negócio na aplicação por meio do ODS. Por exemplo, uma transação de serviço que exija dados atualizados do cliente.*

Situação de Uso – Fábrica
Uma aplicação ERP para inclusão de pedidos é usada pelos Representantes de Atendimento ao Cliente para incluir pedidos a cada hora diretamente no banco de dados de pedidos back-end. Novos pedidos recebidos devem ser transferidos diariamente ao repositório do painel de serviços da empresa. O painel de serviços da empresa fornece à gerência uma visão holística do volume de pedidos a partir do dia útil anterior. A primeira opção pode ser um "envio" diário de novos pedidos da aplicação ERP ao repositório do painel. Ou o painel pode iniciar um "recebimento" do banco de dados de pedidos por meio do Broker de Integração, para fornecer esses dados quando a gerência requisitar a última visualização do volume de pedidos.

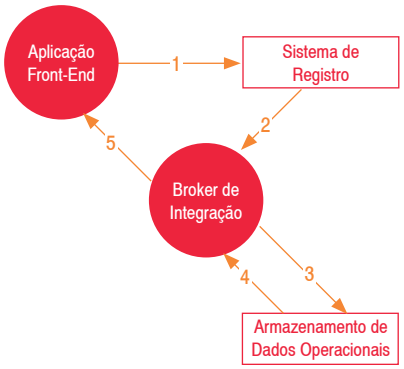
Cada uma dessas opções é explicada com mais detalhes a seguir.

Opção 2a: Envio ao Broker de Integração
A *Figura 3* ilustra a propagação EAI dos registros incrementais fazendo com que o Sistema de Registro envie esses dados ao Broker de Integração. As etapas são executadas nesta seqüência:

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Sistema do Registro notifica o Broker de Integração sobre o recebimento desses dados após a conclusão da atualização.
3. O Broker de Integração recebe essa atualização e envia-a ao ODS.
4. O ODS confirma a atualização.
5. O Broker de Integração envia uma confirmação da propagação bem-sucedida desses dados à aplicação Front-End.

Opção 2b: Recebimento do Broker de Integração
A Figura 4 ilustra a propagação EAI dos registros incrementais fazendo

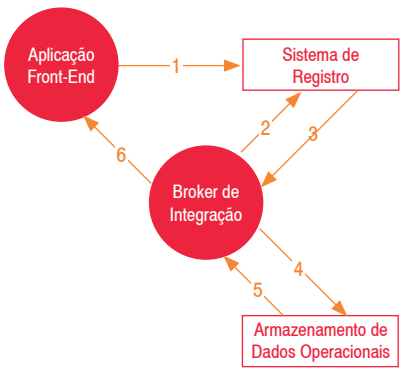
Figura 3. Envio ao Broker de Integração



com que o Broker de Integração consulte o Sistema de Registro regularmente e propague esses dados ao ODS. As etapas são executadas nesta seqüência:

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Broker de Integração consulta o Sistema do Registro para verificar se algum novo dado foi recebido.
3. O Sistema de Registro responde à consulta.
4. Se houver algum novo dado a ser propagado, o Broker de Integração envia uma atualização ao ODS.

Figura 4. Recebimento do Broker de Integração



"Embora os processos de negócio de departamentos não sofram impactos pela proliferação de dados entre vários repositórios, existe uma necessidade de aproveitar os dados no nível empresarial bem como no nível das unidades de negócio."

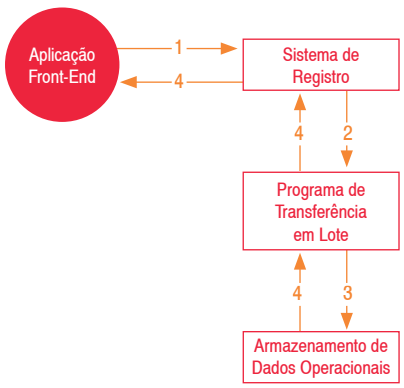
5. O ODS confirma a atualização.
6. O Broker de Integração envia uma confirmação da propagação bem-sucedida desses dados à Aplicação Front-End.

Opção 3: Transferência Incremental em Lotes (Captura de Dados Alterados)

A Opção 3 é orientada a dados e é usada para mover periodicamente dados novos ou alterados da origem para o armazenamento de dados de destino. Esta opção é aplicável a situações em que é aceitável que os dados atualizados no Sistema de Registro sejam fornecidos a outras aplicações após um período finito (por exemplo, um dia). Em tais situações, os dados são transferidos em uma base incremental do Sistema de Registro ao ODS. Esta opção de compartilhamento de dados envolve a captura de dados alterados de uma ou mais aplicações de origem e posterior transporte desses dados a uma ou mais operações de destino em lote. Isso é mostrado graficamente na ***Figura 5***. Considerações típicas nesta opção incluem a identificação da janela de transferência em lote que é propícia nos sistemas de origem e de destino para extrair e transportar os dados. Existem duas formas de fazer isso:

1. Log de Alterações: O Sistema de Registro mantém os dados alterados em conjuntos de registro dedicados de forma que o Programa de Transferência em Lote possa ler diretamente esses conjuntos de registro para obter o delta desde a última transferência. Nesse caso, o Sistema de Registro é responsável pela identificação em tempo real dos dados alterados quando ocorrem alterações.
2. Comparação ao Anterior: O Programa de Transferência em Lote aproveita os dados nos conjuntos de registro base dentro do Sistema de Registro para identificar o conteúdo alterado. Nesse caso, o Programa de Transferência em Lote é responsável pela comparação do estado atual dos dados com estados anteriores para determinar o que mudou nesse ínterim.

Figura 6. Replicação Nativa



A seqüência típica de eventos para esse tipo de compartilhamento de dados é a seguinte:

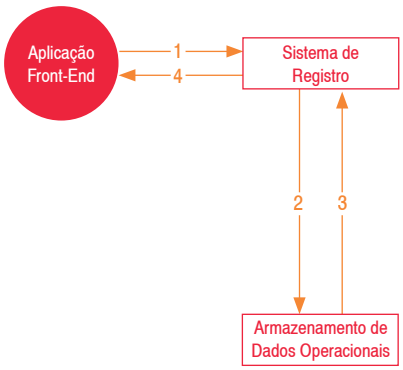
1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Programa de Transferência em Lote coleta os dados alterados do Sistema de Registro.
3. O Programa de Transferência em Lote atualiza o ODS.
4. Uma confirmação é enviada à Aplicação Front-End, ao Sistema de Registro e/ou ao Programa de Transferência em Lote depois da atualização bem-sucedida do ODS.

Situação de Uso – Fornecedor de Serviços

A força de vendas usa um banco de dados de tendências de vendas que rastreia todas as tendências que os representantes de vendas estão buscando. A unidade de entrega de projetos rastreia os recursos necessários para as atividades relacionadas a vendas e entregas. A unidade de entrega de projetos mapeia os requisitos de recursos aos projetos existentes, bem como as tendências em progresso no momento. Para essa finalidade, os dados das tendências são transferidos diariamente do banco de dados de tendência de vendas ao banco de dados de entrega de projetos por meio da opção de transferência incremental em lote.

Opção 4: Replicação Nativa

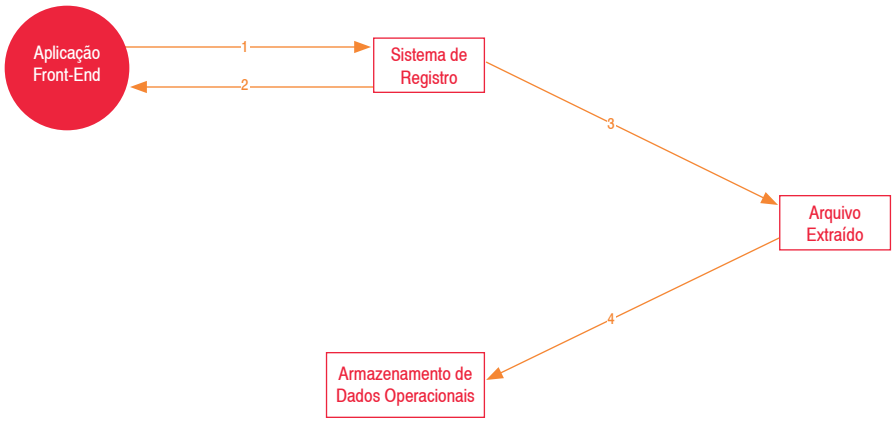
Figura 5. Transferência Incremental em Lotes



A Opção 4 é orientada a dados e é especialmente relevante para situações de alta disponibilidade; por exemplo, serviços de emergência, em que os armazenamentos de dados de origem e destino precisam estar sincronizados virtualmente a todo o momento. Esta opção de compartilhamento de dados envolve o uso de recursos nativos de sistemas de gerenciamento de bancos de dados (DBMS) para refletir alterações em um ou mais bancos de dados de origem para um ou mais bancos de dados de destino. Isso pode ocorrer tanto em tempo real (aproximado) quanto no modo em lote. A seqüência típica de eventos para a replicação nativa é a seguinte:

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. A Replicação Nativa transfere dados do Sistema de Registro para o ODS.
3. O ODS envia uma confirmação do recebimento dos dados de volta ao Sistema de Registro.
4. O Sistema de Registro envia uma confirmação do sucesso da operação à Aplicação Front-End.

Figura 7. Extração de Arquivo



Situação de Uso – Contratante da Assistência Médica
Dados de reivindicações são introduzidos por meio de uma aplicação cliente/servidor de duas camadas no RDBMS back-end pelos Representantes de Atendimento ao Cliente. Atualizações ao Perfil do Cliente também são feitas no Sistema de Registro enquanto são introduzidos dados sobre as reivindicações. As atualizações do Perfil do Cliente são replicadas diretamente no ODS, que serve como o Arquivo de Informações do Cliente para todas as outras aplicações empresariais.

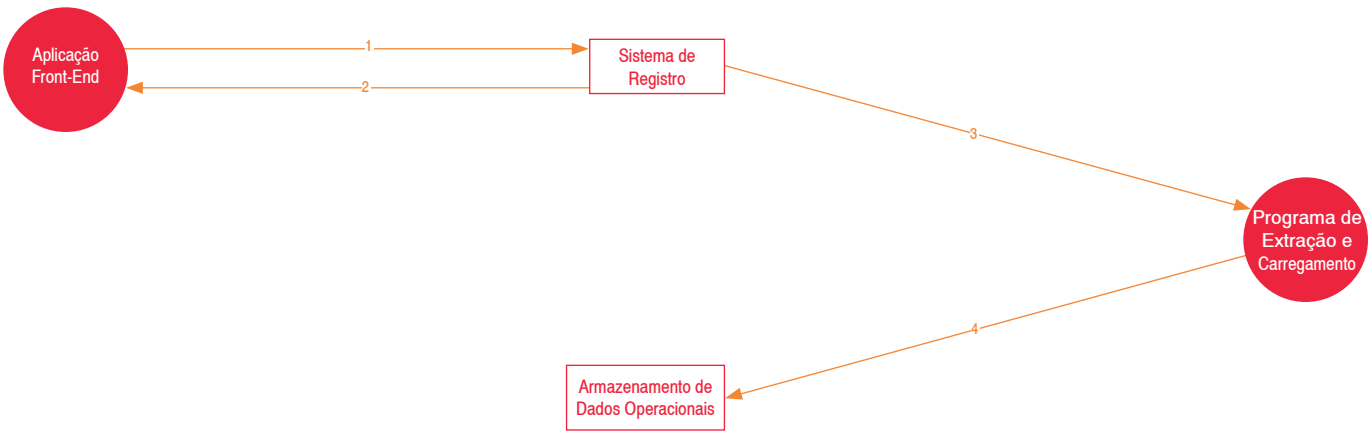
Opção 5: Atualização em Lote
Esta opção é orientada a dados e é apropriada quando grandes quantidades de dados, por exemplo, uma tabela de referências de dados de produtos, precisam ser sincronizadas periodicamente com o Sistema de Registro. Esta opção transfere todos os dados periodicamente, inclusive as alterações mais recentes. Todos os registros são extraídos do Sistema de Registro e atualizados no ODS. Registros existentes no ODS são limpos durante cada transferência. Tais transferências são normalmente realizadas no modo em lote da noite para o dia.

A Atualização em Lote é adequada para situações em que há sobrecarga significativa envolvida na identificação e propagação das alterações incrementais. A abordagem incremental pode ser mais propensa a erros e, portanto, necessitar de muita manutenção. Esses tipos de transferências podem ser realizados de uma das duas maneiras:

- Opção 5a: Extração de Arquivo: Um programa no Sistema de Registro extrai todos os registros em um arquivo intermediário. O arquivo é subseqüentemente carregado no ODS por outro programa.
- Opção 5b: Extração de Programa: Um programa separado consulta o Sistema de Registro e transfere cada registro ao ODS em tempo real. Não é criado qualquer arquivo intermediário.

Opção 5a: Extração de Arquivo com Atualização Completa
A *Figura 7* ilustra o processo de extração baseado em arquivo para a transferência de dados em lote. As seguintes etapas são executadas neste processo:

Figura 8. Extração de Programa



"O modelo de dados em toda a empresa funciona como um banco de dados virtual. Em algum sentido, é uma visão, em termos de bancos de dados relacionais, sobre tabelas espalhadas entre vários bancos de dados físicos."

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Sistema de Registro confirma a atualização.
3. Todos os registros são extraídos em um Arquivo de Extração do Sistema de Registro.
4. O Arquivo de Extração é atualizado no ODS.

Opção 5b: Extração de Programa com Atualização Completa

A *Figura 8* ilustra o processo de extração baseado em programa para a transferência de dados em lote. As seguintes etapas são executadas neste processo:

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Sistema de Registro confirma a atualização.
3. O Programa de Extração e Carregamento recupera e atualiza todos os registros do Sistema de Registro no ODS.

Diferentemente da Extração de Arquivo, a recuperação do Sistema de Registro e as atualizações no ODS fazem parte de uma única transação, sem persistência de dados intermediária. O Programa de Extração e Carregamento pode ser acionado em intervalos fixos ou na ocorrência de eventos específicos. Por exemplo, ele pode ser executado quatro vezes por dia, ou após atualizações em tabelas mestre essenciais. Embora seja semelhante do ponto de vista arquitetônico à Opção 3: Transferência Incremental em Lotes (veja a Figura 5), o escopo é diferente: aqui, todos os dados do Registro do Sistema são transferidos ao ODS, em vez de apenas uma alteração incremental.

Situação de Uso – Departamento de RH de uma Grande Empresa

Grandes empresas internacionais com milhares de funcionários possuem uma hierarquia organizacional que é espalhada ampla e profundamente em todo o mundo. Uma pequena alteração nessa hierarquia pode ter um efeito cascata entre as camadas organizacionais. Embora a estrutura organizacional seja mantida em um único repositório, ela é usada no modo de somente leitura por outras aplicações do ODS. A estrutura organizacional, portanto, deve ser totalmente atualizada regularmente no ODS.

Opção 6: Transferência ETL/ELT

A Opção 6, ilustrada na *Figura 9*, é orientada a dados e é mais apropriada quando uma exclusão ou transformação de dados substanciais é necessária conforme os dados são movidos (por exemplo, para a integração em data warehouse ou data mart). Esta opção sobrepõe-se com a ***Opção 3: Transferência Incremental em Lote*** e a ***Opção 5: Transferência de Atualização em Lote***. A diferença é que a lógica de negócio é aplicada aos dados enquanto são transportados do sistema de origem para o sistema de destino. Uma ferramenta ETL é usada com freqüência para esse tipo de transferência de dados. Os dados de origem são extraídos, transformados no transporte e então carregados em um ou mais bancos de dados de destino. As transformações realizadas nos dados representam as funções de negócio da organização. As funções de negócio garantem que os dados sejam padronizados, limpos e possivelmente aperfeiçoados por meio da agregação ou de outras manipulações antes de serem gravados nos bancos de dados de destino.

A transferência ETL envolve as seguintes etapas:

1. A aplicação front-end inicia a atualização do Sistema de Registro.
2. O Programa de Transferência ETL coleta os dados alterados ou em lote do Sistema de Registro.
3. O Programa de Transferência ETL atualiza o ODS.
4. Uma confirmação é enviada à Aplicação Front-End, ao Sistema de Registro e/ou ao Programa de Transferência ETL depois da atualização bem-sucedida do ODS.

O mesmo aplica-se à transferência ELT. A diferença entre ETL e ELT está no ambiente em que as transformações de dados são aplicadas. No ETL tradicional, a transformação ocorre quando os dados estão trafegando do sistema de origem para o sistema de destino. No ELT, os dados são carregados no sistema de destino e então transformados dentro do ambiente do sistema de destino. Isso se tornou uma opção popular recentemente visto que existem eficiências significativas que podem ser obtidas por meio da manipulação de dados dentro de ambientes de banco de dados (por exemplo, usando stored procedures).

Situação de Uso – Fornecedor de Assistência Médica

Os empregadores enviam informações sobre Habilidades sobre os empregados e seus dependentes para os Contratantes do Seguro de Assistência Médica semanalmente, registrando todas as alterações ocorridas nesse período. Os dados que entram encontram-se em um formato proprietário do Empregador que precisa ser convertido para o formato do sistema mainframe back-end do Fornecedor de Assistência Médica. Registros de resumo precisam ser criados para que listem o número de dependentes e filhos de cada funcionário. As ferramentas ETL

Figura 9. Transferência ETL

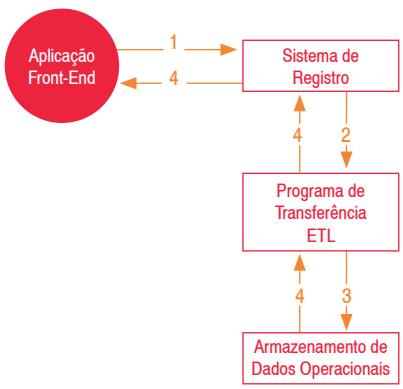
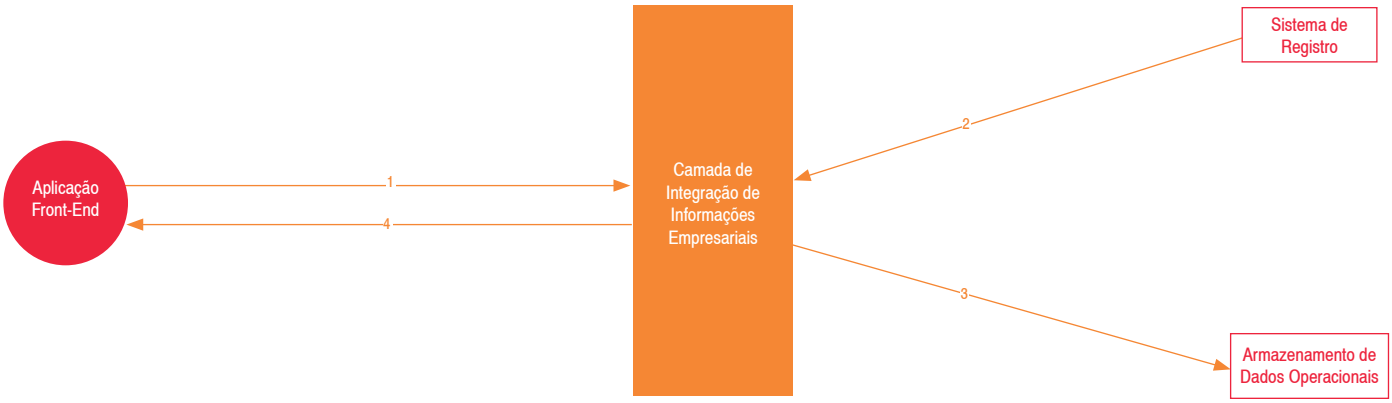


Figura 10. Integração de Informações Empresariais



podem ser usadas para realizar essas transformações de conteúdo e formato no modo em lote.

Opção 7: Integração de Informações Empresariais

Esta opção está em desenvolvimento e é semelhante à Revisão do Processo de Negócio. Ela envolve a criação de um modelo de dados lógico de toda a empresa que representa as principais entidades de negócio e suas relações em uma forma padronizada, consistente. A camada de Integração de Informações Empresariais onde esse modelo encontra-se possui a inteligência de negócio para realizar o seguinte:

- Determinar se o repositório possui o valor mais preciso para cada elemento de dados.
- Construir o conjunto de resultados coletando as informações corretas do repositório correto.
- Propagar informações atualizadas para todos os repositórios afetados de forma que estejam em um estado sincronizado a todo o momento.
- Fornecer uma visão de toda a empresa para todas as entidades de negócio.

O modelo de dados em toda a empresa funciona como um banco de dados virtual. Em algum sentido, é uma visão, em termos de bancos de dados relacionais, sobre tabelas espalhadas entre vários bancos de dados físicos.

Como parte de suas responsabilidades de integração de informações, a camada EII (Enterprise Information Integration) pode propagar as informações ao ODS e ao Sistema de Registro, garantindo que estão sincronizadas. Isso é ilustrado na Figura 10.

As seguintes etapas de execução estão envolvidas quando a opção EII é exercitada:

1. A aplicação front-end inicia a atualização do Sistema de Registro por meio da camada EII.
2. A camada EII atualiza o Sistema de registro.
3. A camada EII atualiza o ODS.
4. Após a conclusão bem-sucedida de ambas as atualizações, a camada EII envia a confirmação de volta para a Aplicação Front-End.

Situações Compostas

Independentemente do Cenário Exemplo descrito no início deste artigo e das situações de uso descritas em cada opção, existem situações complexas em que as várias opções para a transferência de dados precisam ser avaliadas cuidadosamente e uma combinação das opções relevantes precisa ser aplicada. Essas situações incluem, mas não estão limitadas a:

- Preencher um DW ou um ODS com dados de sistemas operacionais
- Preencher data mart de um DW ou de um ODS
- Propagar de volta os dados integrados a aplicações
- Combinações de transferências de dados de aplicação-a-aplicação e aplicação-a-ODS

As três primeiras situações podem ser manipuladas usando a *Revisão do Processo de Negócio e/ou a Opção 1: Transferência EAI em Tempo Real até a Opção 7: Integração de Informações Empresariais*, descritas anteriormente. As situações de aplicação-a-aplicação envolvem uma combinação das opções anteriores e dois tipos são discutidos aqui detalhadamente.

"A opção mais apropriada para um ambiente baseia-se nos requisitos de transferência de dados e nas restrições específicas a esse ambiente."

Opção 8a: Transferência de Aplicação-a- Aplicação com Referência Cruzada
A Opção 8a é apropriada quando a ferramenta EAI deve executar uma busca simples durante a transferência de dados. Por exemplo, enquanto está transferindo dados de uma aplicação de Vendas (X) para uma aplicação de Finanças (Y), o código de conta atual baseado no tipo de transação na transação Vendas deverá ser procurado e adicionado à transação durante a transferência.

O requisito de negócio desta situação, mostrado graficamente na Figura 11, é a transferência de dados da aplicação X para a aplicação Y. Como parte dessa transferência, devem existir manipulações realizadas nos dados que exigem tabelas de referência cruzada (como códigos de busca e conversão em valores significativos no sistema de destino). Enquanto a transferência EAI em tempo real pode efetuar a transferência de dados da aplicação X para a aplicação Y, a transferência ETL pode ser usada para transferir dados de referência cruzada desses sistemas a uma construção de dados de referência cruzada (representados por XREF no diagrama).

Observação: *A Opção 5a: Extração de Arquivo com Atualização Completa ou a Opção 5b: Extração de Programa com Atualização Completa também pode ser usada para atualizar a tabela XREF.*

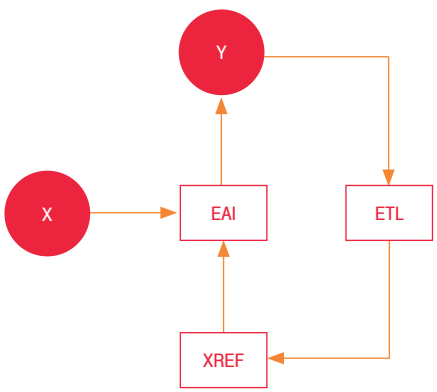
Opção 8b: Transferência de Aplicação-a- Aplicação com Dados Estáticos
A Opção 8b representa uma situação em que os dados da aplicação X devem ser complementados com os dados da aplicação Z durante a transferência à aplicação Y. Por exemplo, uma transação da aplicação de Vendas (X) deve ser complementada pelos dados de custo de produto da aplicação de Inventário (Z) durante a transferência à aplicação de Finanças (Y).

Nesta situação, ilustrada na Figura 12, os dados são transferidos da aplicação X à Y. Ao mesmo tempo, a atualização da aplicação Y também envolve o recebimento de outros dados de aplicações secundárias estáticas – ou pelo menos relativamente estáticas comparadas à natureza de transferência em tempo real da transferência de X para Y. Aqui, o EAI é usado para chegar à transferência de algum dado de X para Y. A transferência ETL é usada para preparar e fornecer os dados adicionais que a aplicação Y requer da aplicação secundária (Z) no ODS. O EAI coleta, então, os dados adicionais do ODS para preencher a aplicação Y.

Observação: *qualquer uma das opções, seja a Opção 6: Transferência ETL/ELT ou a Opção 7: Integração de Informações Empresariais pode ser usada para atualizar o ODS.*

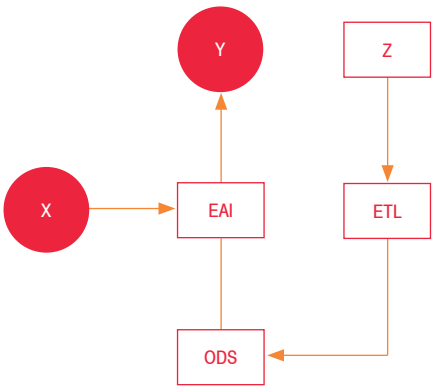
Análise de Opções
A opção mais apropriada para um ambiente baseia-se nos requisitos de transferência de dados e nas restrições específicas a esse ambiente. Existem vários critérios de procedimento,

Figura 11. Transferência de Aplicação-a-Aplicação com Referência Cruzada



financeiros e de arquitetura que devem ser levados em consideração ao se determinar a opção mais adequada para algum ambiente. Esta seção resume os principais critérios a serem considerados, seguidos pela classificação de cada opção no contexto desses critérios. Embora possa haver outras opções aplicáveis ou combinações dessas opções, conforme discutido em Situações Compostas, esta seção concentra-se nas opções básicas (1 até 6), descritas anteriormente.

Figura 12. Transferência de Aplicação-a-Aplicação com Dados Estáticos



A Revisão do Processo de Negócio e a Integração de Informações Empresariais foram excluídas da análise, visto que não envolvem realmente a transferência de dados.

Esses critérios podem ser classificados em Requisitos e Limitações, conforme mostrado na Tabela 1. Os requisitos são tipicamente arquitetônicos, orientados pelas necessidades de negócio. As restrições definem os parâmetros dentro dos quais a solução deve ser projetada tendo a implementação geral e o esforço de manutenção em mente.

Tabela 1. Critérios de Avaliação

Critério	Categoria	Descrição
Latência	Requisito	Qual a velocidade da transferência de dados?
Transformação	Requisito	Complexidade da transformação a ser executada como parte da transferência
Volume	Requisito	Quantidade de dados trocados durante cada transferência
Invasão	Restrição	Grau de alteração em aplicações existentes para efetuar a transferência de dados
Esforço	Restrição	Esforço necessário para criar e manter a solução

Tabela 2. Avaliação das Opções

Critério / Opção	1 – Transferência EAI em Tempo Real	2 – Propagação EAI de Registros Incrementais	3 – Transferência Incremental em Lotes	4 – Replicação Nativa	5 – Atualização em Lote com Transferência em Lote	6 – ETL/ELT
Latência	Tempo Real Aproximado	Tempo Real Aproximado	Lote	Tempo Real Aproximado	Lote	Lote
Transformação	Alta	Alta	Média	Não Aplicável	Não Aplicável	Alta
Volume	Baixo	Baixo	Médio	Baixo	Alto	Alto
Invasão	Alta	Média	Alta	Baixa	Baixa	Baixa
Esforço	Alto	Mádio	Médio	Baixo	Baixo	Alto

A *Tabela 2* resume as características da *Opção 1: Transferência EAI em Tempo Real* até a *Opção 6: Transferência ETL/ELT* no contexto desses critérios.
Observe que a *Revisão do Processo de Negócio e a Opção 7: Integração de Informações Empresariais* não foram analisadas na Tabela 2. A Revisão do Processo de Negócio é uma revisão dos processos de negócio existentes que pode resultar na implementação de qualquer uma das outras opções. A Opção 7: Integração de Informações Empresariais está relacionada à representação lógica das informações no nível empresarial. Qualquer uma das opções, da *Opção 1: Transferência EAI em Tempo Real até a Opção 6: Transferência ETL/ELT*, pode ser usada em conjunto com o modelo EII.

E G Nadhan
Chefe, EDS
Easwaran.Nadhan@eds.com

E G Nadhan é Chefe do grupo de Integração Empresarial Estendida da EDS. Com mais de 20 anos experiência na indústria do software, Nadhan é responsável pela entrega de soluções EAI e B2B integradas a clientes de larga escala.

Conclusão
Existem várias abordagens disponíveis para que as empresas efetuem a transferência de dados entre suas aplicações de negócio. As empresas devem, primeiramente, revisar o Processo de Negócio para confirmar a necessidade da transferência. Depois de confirmada, existem várias opções, habilitadas pelas tecnologias EAI e ETL, disponíveis para efetuar a transferência de dados. Em alguns casos, uma combinação dessas opções pode ser necessária para atender aos requisitos de transferência de dados de conjuntos completos dentro de uma empresa. O processo que conduz tais transferências deve estabelecer a tecnologia e a ferramenta empregada, em vez de a tecnologia definir o processo. Grandes empresas normalmente empregam uma mistura melhorada dessas três estratégias: Revisão do Processo de Negócio, EAI e ETL. A Integração de Informações Empresariais está se desenvolvendo como outra opção disponível nesse espaço. A opção correta ou combinação de opções a ser usada para uma determinada situação depende de vários critérios, alguns dos quais são orientados por requisitos e outros são restrições. Este artigo apresenta os critérios mais significativos a serem considerados e fornece uma avaliação de cada opção com base nesses critérios. *Agradecimento Especial*:

Os autores agradecem a Carleen Christner, Consultora-Gerente do grupo de Integração Empresarial Estendida da EDS por sua revisão completa do artigo e pelos comentários fornecidos sobre o conteúdo e o formato.

Jay-Louise Weldon
Consultor-Gerente, EDS
Jaylouise.weldon@eds.com

Jay-Louise Weldon é Consultor-Gerente do grupo de Serviços de Inteligência de Negócio da EDS. Jay-Louise tem mais de 20 anos de experiência em soluções de inteligência de negócio e em design de bancos de dados e sistemas.

Messaging Patterns na Arquitetura Orientada a Serviços – Parte 2

Por Soumen Chatterjee

Introdução
Na primeira parte deste artigo publicado na edição 2 do JOURNAL, descrevemos como os Patterns de mensagens existem em níveis diferentes de abstração na SOA. Especificamente, os Patterns de Tipos de Mensagens foram usados para descrever uma variedade de mensagens na SOA, os Patterns de Canais de Mensagens explicaram os sistemas de transporte de mensagens e, finalmente, os Patterns de Roteamento explicaram os mecanismos para rotear mensagens entre o Fornecedor e o Consumidor dos Serviços. Nesta segunda parte, iremos abordar os Patterns de Contrato, que ilustram as especificações comportamentais necessárias para manter uma comunicação suave entre o Fornecedor e o Consumidor de Serviços, e os Patterns de Construção de Mensagens, que descrevem a criação do conteúdo da mensagem que percorre o sistema de mensagens.

Contratos e Ocultação de Informações
Um contrato de interface é um acordo publicado entre o fornecedor e o consumidor de serviços. O contrato especifica não só os argumentos e os valores de retorno que o serviço oferece, mas também as pré e pós-condições desse serviço.

Parnas e Clements descrevem muito bem os princípios da ocultação de informações:

"Nossa estrutura de módulo baseia-se no critério de decomposição conhecido como ocultação de informações [IH]. Segundo esse princípio, os detalhes do sistema que provavelmente alteram-se de forma independente devem ser os segredos de módulos separados; as únicas suposições que devem aparecer nas interfaces entre os módulos são aquelas que possivelmente não se alterarão. Cada estrutura de dados é usada em apenas um módulo; um ou mais programas dentro do módulo podem acessá-la diretamente. Qualquer outro programa que exigirexija informações armazenadas em estruturas de dados do módulo deve obtê-las solicitando o acesso aos programas que pertencem a esse módulo".
(Parnas e Clements 1984)[8]

Se essa afirmação for aplicada à SOA, os serviços nunca deverão expor suas estruturas de dados internas. Do contrário, causa dependências desnecessárias (agrupamento estreito) entre o fornecedor de serviços e seus consumidores. Os detalhes da implementação interna são expostos por meio da criação de um design de interface parametrizado mapeado aos aspectos de implementação do serviço, em vez dos aspectos funcionais.

Pattern de Contrato
Problema:
Como os comportamentos podem ser definidos independentes de implementações?

Solução:
O conceito de contrato de interface foi adicionado a linguagens de programação, como C# e Java, para descrever o comportamento na sintaxe e na semântica. A semântica interna dos dados deve ser mapeada para a semântica externa de um contrato independente. O contrato depende somente do domínio do problema da interface, e não dos detalhes de implementação.

Interações:
Os métodos, os tipos de métodos, os tipos dos parâmetros dos métodos e os tipos de campo determinam a sintaxe da interface. Os comentários, os nomes dos métodos e os nomes dos campos descrevem a semântica da interface. Um objeto pode implementar várias interfaces.

Construção da Mensagem
A mensagem em si própria é simplesmente algum tipo de estrutura de dados – como strings, matrizes de bytes, registros ou objetos. Ela pode ser interpretada simplesmente como dados, como a descrição de um comando a ser chamado pelo destinatário, ou como a descrição de um evento ocorrido no remetente. Quando duas aplicações desejam trocar dados, elas o fazem ocultando-os na mensagem. A construção da mensagem introduz os problemas de design a serem considerados após a geração da mensagem. Nesse catálogo de Patterns de construção de mensagem, iremos apresentar três Patterns de construção de mensagens importantes.

Identificador de Correlação
Problema:
Em qualquer sistema de mensagens, o consumidor deve enviar várias solicitações de mensagens a diferentes fornecedores de serviços. Como resultado disso, ele recebe várias respostas. Deve haver algum mecanismo para correlacionar as respostas às solicitações originais.

Solução:
Cada mensagem de resposta deve conter um identificador de correlação, uma ID exclusiva que indica a qual mensagem de solicitação a resposta destina-se. Essa ID de correlação é gerada com base em uma ID exclusiva contida dentro da mensagem de solicitação.

Interações:
Existem seis partes no Identificador de Correlação:
- **Solicitador** – Aplicação do consumidor.
- **Replicador** – Fornecedor de Serviços. Recebe a solicitação de ID e armazena-a como a ID de correlação na resposta.
- **Solicitação** – Mensagem enviada do Consumidor para o Fornecedor de Serviços que contém a ID de solicitação.

"Os serviços nunca deverão expor suas estruturas de dados internas."

Figura 25: Identificador de Correlação

– ***Resposta*** – Mensagem enviada do Fornecedor de Serviços para o Consumidor que contém a ID de correlação.
– ***ID de Solicitação*** – Um token na solicitação que identifica exclusivamente a solicitação.
– ***ID de Correlação*** – Um token na resposta que tem o mesmo valor que a ID de solicitação na solicitação.

Mecanismo de Funcionamento:
Durante o momento da criação, a mensagem de solicitação é atribuída com uma ID de solicitação. Quando o fornecedor de serviços processa a solicitação, ele salva a ID de solicitação e adiciona essa ID à resposta como uma ID de correlação. Dessa forma, ajuda a identificar a correspondência daentre solicitação- e resposta. Uma ID de correlação (e também a ID de solicitação) é geralmente associada ao cabeçalho em vez do corpo da mensagem e pode ser tratada como um metadado da mensagem.

Seqüência da Mensagem

Problema:
Por causa da natureza distribuída inerente da mensagem, a comunicação geralmente ocorre pela rede. É necessário utilizar a largura de banda apropriada, mantendo o melhor desempenho. Em determinadas situações (como o envio de uma lista de faturas para algum cliente em particular), talvez precise enviar grandes quantidades de dados (100 MB ou mais). Em tais casos é recomendável dividir os dados em quantidades menores e então enviá-los como um conjunto de mensagens. O problema é como reorganizar as quantidades de dados para formar o conjunto completo.

Solução:
Use uma seqüência de mensagens, e marque cada mensagem com campos de identificação da seqüência.

Interação:
Os três campos de identificação da seqüência da mensagem são:
– ***ID da Seqüência*** – Usada para diferenciar uma seqüência de outra.
– ***ID da Posição*** – Uma ID relativa exclusiva para identificar a posição da mensagem dentro de uma seqüência particular.
– ***Final do Indicador da Seqüência*** – Usado para indicar o final de uma seqüência.

Normalmente, as seqüências são criadas de forma que cada mensagem na seqüência indique o tamanho total da seqüência, ou seja, o número de mensagens na seqüência (veja a *Figura 26*).

Como uma alternativa, é possível criar seqüências de forma que cada mensagem indique se é a última mensagem na seqüência (veja a *Figura 27*).

Utilizemos um exemplo da vida real. Suponha-se que desejamos gerar um relatório para todas as faturas de 01/01/2001 até 31/12/2003. Isso pode resultar em milhões de registros. Para cuidar dessa situação, divida o cronograma em trimestres e retorne dados para cada trimestre. O remetente envia os dados trimestrais como mensagens e o destinatário usa o número da seqüência para remontar os dados e identificar a conclusão dos dados recebidos, com base no indicador de Final de Seqüência.

Vencimento da Mensagem

Problema:
As mensagens são armazenadas em disco ou em mídia persistente. Com o número crescente de mensagens, o espaço em disco é consumido. Ao final do ciclo de vida das mensagens, elas devem expirar e ser destruídas para recuperar o espaço em disco.

Figura 26: Seqüência da Mensagem indicando o tamanho

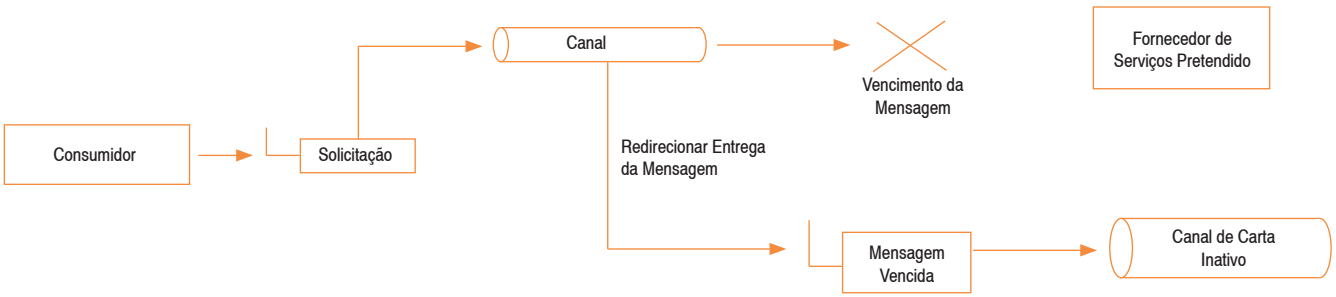
Seqüência no. 1	Seqüência	1	Seqüência	1	Seqüência	1
	Posição	1	Posição	2	Posição	n
	tamanho	1	tamanho	n	tamanho	n
	Corpo da Mensagem		Corpo da Mensagem		Corpo da Mensagem	

Figura 27: Seqüência da Mensagem com indicador de final de mensagem

Seqüência no. 9	Seqüência	9	Seqüência	9	Seqüência	9
	Posição	0	Posição	1	Posição	n°1
	Final	F	Final	F	tamanho	T
	Corpo da Mensagem		Corpo da Mensagem		Corpo da Mensagem	

"A construção da mensagem introduz os problemas de design a serem considerados após a geração da mensagem. As diferenças do remetente e do destinatário com relação ao formato da mensagem são conciliadas pela transformação da mensagem."

Figura 28: Vencimento da Mensagem



Solução:
Configure o vencimento da mensagem para especificar um limite de tempo para a preservação das mensagens em mídias persistentes.

Interação:
O vencimento da mensagem é uma gravação de data e hora que decide o período de existência da mensagem.

Quando alguma mensagem vence, o sistema de mensagens simplesmente descarta-a ou transfere-a para um canal de dead-letter.

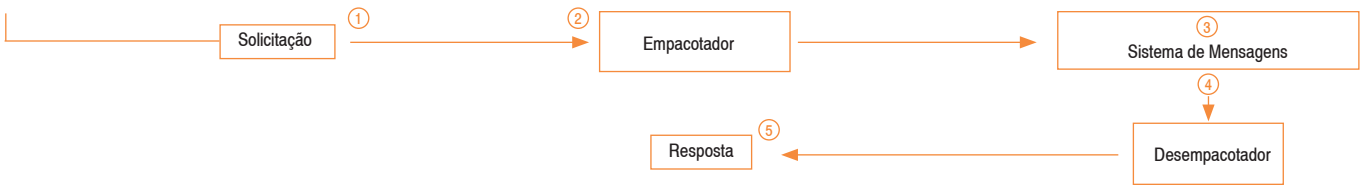
Transformação da Mensagem
Várias aplicações podem não entrar em acordo com relação ao formato para a mesma data conceitual; o remetente formata a mensagem de uma forma, mas o destinatário espera que esteja formatada de outra maneira. Para conciliar essa situação, a mensagem deve passar por um procedimento intermediário de conversão que transforma a mensagem de um formato para outro. A transformação da mensagem pode envolver alteração em dados (adição, remoção permanente ou temporária de dados) em nós existentes, por meio da implementação de regras de negócio. Algumas vezes pode também enriquecer um nó vazio. Aqui, apresentaremos alguns Patterns importantes de transformação de mensagens.

Empacotador de Envelope
Problema:
Quando um formato de mensagem é Quando um formato de mensagem é encapsulado dentro de outro, o sistema pode não conseguir acessar os dados do nó. A maioria dos sistemas de mensagens permite que componentes (por exemplo, um roteador baseado em conteúdo) acessem somente campos de dados que façam parte do cabeçalho definido da mensagem. Se uma mensagem for empacotada em um campo de dados dentro de outra mensagem, o componente pode não conseguir usar os campos para realizar o roteamento ou a transformação baseada em regras de negócio. Portanto, alguns campos de dados podem ser elevados da mensagem original para o cabeçalho da mensagem do novo formato de mensagem.

Soluções:
Use um empacotador de envelope para empacotar os dados dentro de um envelope que esteja em conformidade com a infra-estrutura da mensagem. Desembale a mensagem quando chegar ao destino.

Interações:
O processo de embalar e desembalar a mensagem consiste em cinco etapas:
1. A fonte da mensagem publica uma mensagem dependente de um formato bruto.
2. O embalador pega a mensagem bruta e a transforma em um formato que esteja em conformidade com o sistema de mensagens. Isso pode incluir a adição de campos de cabeçalho de mensagem, a criptografia da mensagem, a adição de credenciais de segurança etc.
3. O sistema de mensagens processa as mensagens que estão em conformidade.

Figura 29: Empacotador de Envelope



4. A mensagem resultante é entregue ao desempacotador. O desempacotador desfaz qualquer modificação feita pelo empacotador. Isso pode incluir a remoção de campos de cabeçalho, a descriptografia da mensagem ou a verificação das credenciais de segurança.
5. O consumidor recebe uma mensagem com "texto simples".

Um envelope normalmente embala o cabeçalho da mensagem e o corpo ou a carga da mensagem. Podemos considerar o cabeçalho como as informações fora do envelope – elas são usadas pelo sistema de mensagens para rotear e rastrear a mensagem. O conteúdo do envelope é a carga ou o corpo da mensagem – a infra-estrutura da mensagem não se importa com isso até que chegue ao seu destino.

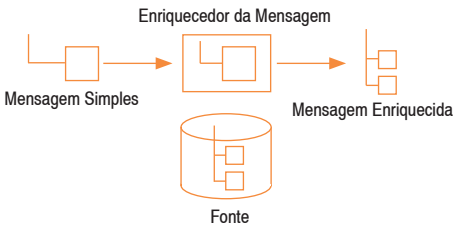
Enriquecedor de Conteúdo

Problema:
Consideremos este exemplo. Um sistema de processamento de empréstimos on-line recebe informações incluindo o número do cartão de crédito do cliente e um Número de Previdência Social. Para concluir o processo de aprovação, é necessário executar uma verificação completa do histórico de crédito. No entanto, esse sistema de processamento de empréstimos não possui os dados do histórico de crédito. Como poderemos nos comunicar com outros sistemas se o originador da mensagem não possui todos os campos de dados disponíveis?

Solução:
Use um transformador especializado, um enriquecedor de conteúdo, para acessar a fonte de dados externa para enriquecer a mensagem com as informações faltantes.

Interações:
O enriquecedor de conteúdo usa as informações incorporadas na mensagem recebida para recuperar dados da fonte externa. Após a recuperação bem-sucedida dos dados necessários da fonte, esses dados são anexados à mensagem.

Figura 30: Enriquecedor de Conteúdo

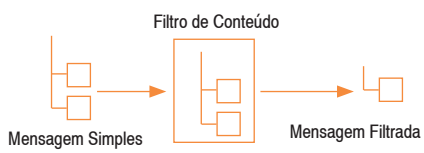


O enriquecedor de conteúdo é usado em várias ocasiões para resolver IDs de referência contidas nas mensagens. Para que as mensagens sejam pequenas, gerenciáveis e fáceis de transportas, muito freqüentemente apenas passamos referências simples de objetos ou chaves em vez de passar o objeto completo com todos os elementos de dados. O enriquecedor de conteúdo recupera os dados necessários com base nas referências de objetos incluídas na mensagem original.

Filtro de Conteúdo

Problema:
O enriquecedor de conteúdo ajuda em situações quando algum destinatário da mensagem requer mais (ou diferentes) elementos de dados do que os contidos na mensagem original. Existem várias situações em que a reversão é desejada, ou seja, a remoção dos elementos de dados da mensagem. O motivo por trás dessa remoção é simplificar a manipulação das mensagens, remover os dados de segurança confidenciais e reduzir o tráfego de rede. Por isso, precisamos simplificar os documentos de origem para incluir somente os elementos que realmente nos interessam.

Figura 31: Filtro de Conteúdo



Solução:
Use um filtro de conteúdo para remover da mensagem itens de dados sem importância.

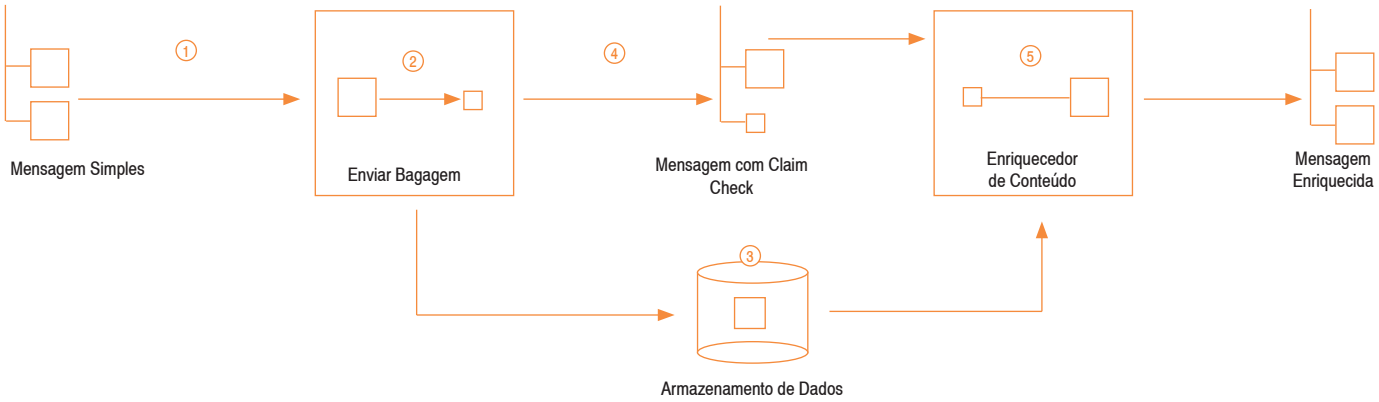
Interações:
O filtro de conteúdo não só remove os elementos de dados, mas também simplifica a estrutura da mensagem. Várias mensagens originadas de sistemas externos contêm vários níveis de grupos aninhados e repetidos, por serem modelados com base em estruturas de bancos de dados normalizadas e genéricas. O filtro de conteúdo nivela essa hierarquia complexa de mensagens aninhadas. Vários filtros de conteúdo podem ser usados para quebrar uma mensagem complexa em mensagens individuais que lidam, cada uma, com um determinado aspecto da mensagem grande.

Claim Check

Problema:
O *enriquecedor de conteúdo* enriquece os dados da mensagem e o *filtro de conteúdo* remove itens de dados desnecessários da mensagem. Algumas vezes, no entanto, essa situação pode ser um pouco diferente. *A transferência de grandes quantidades de dados pelas mensagens pode ser ineficiente devido às limitações na rede ou ao tamanho da mensagem; por isso, podemos precisar remover temporariamente os campos para evitar etapas de processamento específicas que não são necessárias, e então adicioná-los de volta à mensagem, em um momento posterior.*

Solução:
Armazene os dados da mensagem em um armazenamento persistente e passe um claim check aos componentes subseqüentes. Esses componentes podem usar o claim

Figura 32: Claim Check



check para recuperar as informações armazenadas usando o enriquecedor de conteúdo.

Interações:
O pattern Claim Check consiste nas cinco seguintes etapas:

1. Chega uma mensagem com dados.
2. O componente "enviar bagagem" gera uma chave exclusiva que é usada em um estágio posterior como o envio de solicitação.
3. O componente de envio da bagagem extrai os dados com base na chave exclusiva do armazenamento persistente.
4. Os dados persistentes são removidos da mensagem e adicionados ao envio de solicitação.
5. Os dados enviados são recuperados por meio do enriquecedor de conteúdo para recuperar os dados com base no envio de solicitação.

O processo é análogo ao envio de bagagem em aeroportos. Se não desejar levar sua bagagem com você, simplesmente deixe-a no balcão da companhia aérea. Em retorno você receberá uma etiqueta em seu bilhete que possui o número de referência que identifica exclusivamente cada bagagem enviada. Após chegar ao destino final, é possível recuperar sua bagagem.

Conclusão
A SOA destaca a interoperabilidade, que é a capacidade de se comunicar com diferentes plataformas e linguagens. As empresas precisam de uma solução neutra de tecnologias para orquestrar os processos de negócio entre as verticais. A SOA, então, apresenta uma mudança do paradigma tradicional de EAI (Integração de aplicações empresariais), em que a automação de processos de negócio necessitava de conectividade especifica entre aplicações. Segundo Robert Shimp, vice-presidente de Marketing de Tecnologia na Oracle:

"A EAI exige conhecimento específico sobre o que cada aplicação forneceu antecipadamente. A SOA considera cada aplicação um fornecedor de serviços e permite a verificação dinâmica de serviços por meio de um diretório comum, o UDDI de serviços web (Universal Description Discovery and Integration – Descoberta e Integração de Descrição Universal)." [10]

As mensagens são as bases da SOA. Steven Cheah, diretor de engenharia e arquitetura de software na Microsoft Cingapura, declara:

"Agora, temos finalmente um veículo padrão para chegar à SOA. Agora poderemos definir os Patterns de mensagens para SOA usando esses Patterns de serviços web."[10]

Cheah considera a SOA "um aperfeiçoamento da EAI". Especificamente, a SOA recomenda alguns princípios, que realmente ajudam a obter uma melhor integração de aplicações. Esses princípios incluem a descrição de serviços pelas funções que desempenham; a apresentação de serviços como funções flexíveis com os detalhes de seus trabalhos internos invisíveis para outros que desejarem utilizá-los; o uso de mensagens como a única via de "entrada" e "saída" dos serviços; e o controle federado da SOA entre domínios organizacionais, sem que ninguém tenha o controle total dela.

Iniciamos no nível mais alto, com uma visão da empresa orientada por serviços. Descemos então pela arquitetura comum (SOA) e continuamos com a descrição das mensagens. Agora, estamos armados com Messaging Patterns valiosos para atacar as complexidades da SOA e para obter a visão da empresa de serviços orientada a processos dinâmicos.

"A SOA é 'um aperfeiçoamento da EAI'. Especificamente, a SOA recomenda alguns princípios, que realmente ajudam a obter uma melhor integração de aplicações."

Referências

1. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Gregor Hohpe e Bobby Woolf, Addison-Wesley, 2004
2. *Service Oriented architecture: A Primer*, Michael S Pallos, EAI Journal, dezembro de 2001
3. *Solving Information Integration Challenges in a Service-Oriented Enterprise*, ZapThink Whitepaper, http://www.zapthink.com
4. *SOA and EAI*, De Gamma Website, http://www.2gamma.com/en/produit/soa/eai.asp
5. *Introduction to Service-Oriented Programming*, Guy Bieber e Jeff Carpenter, Project Openwings, Motorola ISD, 2002
6. *Java Web Services Architecture*, James McGovern, Sameer Tyagi, Michael Stevens e Sunil Mathew, Morgan Kaufman Press, 2003
7. *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*, Alan Brown, Simon Johnston e Kevin Kelly, IBM, junho de 2003
8. *The Modular Structure of Complex Systems*, Parnas D e Clements P, IEEE Journal, 1984
9. *Design Patterns: Elements of Reusable Object-Oriented Software*, Gamma E, Helm R, Johnson R e Vlissides J, Addison-Wesley, 1994
10. *Computerworld Year-End Special: 2004 Unplugged*, Vol. 10, Edição No. 10, 15 de dezembro de 2003 - 6 de janeiro de 2004, http://www.computerworld.com.sg/pcwsg.nsf/currentfp/fp
11. *Applying UML and Patterns – An introduction to OOA/D and the Unified Process*, Craig Larman, 2001

Soumen Chatterjee, schatterjee@ieee.org

Soumen é um Profissional Certificado da Microsoft e um Arquiteto Empresarial Certificado da Sun. Ele está envolvido significativamente na integração de aplicações empresariais e no desenvolvimento de sistemas distribuídos orientados por objetos por meio da tecnologia Java/J2EE, para servir gigantes globais nas indústrias de finanças e da assistência médica. Com experiência em padrões de design EAI, messaging patterns e estratégias de testes, ele faz designs e desenvolve arquiteturas EAI escaláveis, reutilizáveis, sintonizadas à manutenção e ao desempenho. Soumen é admirador da metodologia de programação extrema e seus principais interesses são AOP e EAI. Além de softwares, ele gosta de filmes, músicas e segue tecnologias de poder da mente.

JOURNAL3

Microsoft®
Microsoft é marca registrada da Microsoft Corporation.