Aprenda a disciplina, busque a arte e contribua com idéias no endereço www.ArchitectureJournal.net
**Recursos com os quais você pode contar.**

# THE ARCHITECTURE JOURNAL™

Idéias para melhores resultados

Jornal 6

# Estratégias para projetos



Microsoft

# Sumário

THE ARCHITECTURE JOURNAL
Idéias para melhores resultados

Jornal 6

# Apresentação

## Caro arquiteto,

No início, eu gostaria de ter publicado apenas seis edições do *O Jornal de Arquitetura* e obter provas suficientes para ajudar a Microsoft a decidir se valia ou não à pena continuar publicando. Os leitores mais freqüentes sabem o que aconteceu com *O Jornal de Arquitetura*: foi algo maravilhoso para todos nós, incluindo nossas super-estrelas – os autores –, ver que o interesse e as assinaturas aumentaram num ritmo impressionante no mundo todo, mês após mês, tanto no formato digital quanto no formato impresso. Foi algo fantástico!

Agora chegou a hora de uma equipe profissional da sede corporativa da Microsoft assumir esta publicação e desenvolvê-la além dos meus sonhos iniciais. Como componente essencial do amplo programa da Microsoft para arquitetos, faz todo o sentido *O Jornal de Arquitetura* receber melhores recursos em termos financeiros e de pessoal, mais do que eu poderia gerenciar com a minha equipe na Europa. Para esta finalidade, fico muito contente de apresentar Simon Guest, gerente de programa de grupos da Equipe de Estratégia de Arquitetura Microsoft, como o novo editor do *O Jornal de Arquitetura*. Simon e sua equipe têm uma ótima visão e uma grande paixão por esta publicação, e tenho certeza de que vão se manter fiéis ao seu princípio original como plataforma independente para os livres pensadores e profissionais da arquitetura de TI.

Nesta edição, temos uma série de ótimos artigos que investigam a governança SOA, a modelagem prática de informações que usa mapas de tópicos, fábricas de softwares para protocolos de colaboração na indústria de assistência médica, e uma perspectiva orientada ao serviço sobre inteligência comercial. Também temos dois documentos curtos sobre modelagem de softwares e planejamento técnico. Muitos desses documentos discutem minhas áreas favoritas.

Vou continuar a participar do *O Jornal de Arquitetura* e estou ansioso para ver Simon e sua equipe perceberem todo o potencial do jornal. Desejo a eles todo o sucesso neste desafio.

Gostaria de agradecer todos vocês pelo grande apoio e estímulo que me demonstraram pessoalmente. Pelos autores pioneiros do *O Jornal de Arquitetura* só tenho a dizer sobre o meu mais profundo respeito e gratidão. Obrigada. Boa sorte!

Arvindra Sehmi

# Levando a governança ao limite

por Philip Boxer e Richard Veryard

**Resumo**

Descubra os desafios enfrentados pelas formas assimétricas de governança e uma abordagem de trabalho que se foca particularmente na forma que uma empresa compreende os riscos que surgem do modo como ela se relaciona com geometrias exógenas, além dos riscos mais conhecidos associados à gestão de sua geometria endógena.

Há duas visões concorrentes sobre SOA (Arquitetura Orientada ao Serviço) circulando na indústria de software, as quais podemos denominar SOA 1.0 e SOA 2.0 (consulte a Tabela 1). Nossa abordagem da governança destina-se à SOA 2.0. Uma das questões centrais levantadas por Christopher Alexander no seu mais recente trabalho de quatro volumes, "The Nature of Order", é como obter ordem sem impor um planejamento (central) de cima para baixo ou, do contrário, como permitir e estimular a inovação de baixo para cima sem perder a ordem (consulte Recursos). Alexander promove o conceito das transformações que preservam a estrutura e argumenta que, sob certas condições, a ordem em ampla escala pode emergir de um processo revolucionário de pequenas etapas incrementais, desde que cada etapa apresente uma preservação apropriada de estrutura. No entanto, como (e em nome de quem) definimos a estrutura que está sendo conservada e em que nível?

Observe que, com esse conceito de preservação de estrutura, a liderança não está fazendo o projeto, mas definindo os parâmetros nos quais o projeto ordenado pode acontecer. Essa função é muito mais um papel de governança. Para responder à questão, precisamos diferenciar governança de gerenciamento.

É possível deixar a discussão sobre governança fluir (tanto a governança de TI em geral quanto a governança SOA em particular) até que pareça cobrir todos os aspectos do gerenciamento. Assim, onde termina o gerenciamento e começa a governança?

Se compreendermos o gerenciamento como algo que visa a fazer o que precisa ser feito, então podemos compreender melhor a governança como a direção (por exemplo, o que faz o comitê diretor). Antigamente, era prática comum para os grandes projetos/programas de desenvolvimento ter comitês diretores – em geral, um para cada projeto – que controlavam coisas como financiamento, escopo, direção e prioridades. O comitê diretor era o fórum ao qual o gerente de projeto se reportava. Seu objetivo era manter um equilíbrio adequado entre os interesses do projeto e os interesses de toda a organização, e resolver os conflitos.

No entanto, havia vários problemas com essa abordagem. Em primeiro lugar, o comitê diretor se encontrava, normalmente, com pouca freqüência, e só tinha uma vaga idéia do que estava acontecendo. Em segundo lugar, os comitês diretores geralmente não conversavam uns com os outros. Em terceiro lugar, havia muitas funções de TI importantes (como a arquitetura) e resultados desejados (como produtividade e reutilização) que não tinham um comitê diretor. Assim, a abordagem do comitê diretor era incompleta e inconsistente e geralmente produzia resultados abaixo do esperado.

## Governança em tempo real

Logo a moda dos comitês diretores ad hoc passou, e muitas organizações grandes de TI começaram a dar uma atenção explícita às questões da governança de TI. Quando muitas atividades de TI eram terceirizadas, essa governança incluía questões de compra e relacionamento com os principais fornecedores.

Com a SOA, temos uma nova abordagem para fazer as coisas. Também temos todos os antigos problemas de governança, além de alguns problemas novos. Há uma gama complexa de atividades orientadas a serviços acontecendo, e todas elas precisam ser adequadamente coordenadas e alinhadas com as metas, e a escala de tempo mudou. Em vez de os comitês diretores se encontrarem a cada dois meses, temos uma governança em tempo real, abrangendo a governança de projeto e a governança em tempo de execução.

> "COM A SOA, TEMOS UMA NOVA ABORDAGEM PARA FAZER AS COISAS. TAMBÉM TEMOS TODOS OS ANTIGOS PROBLEMAS DE GOVERNANÇA, ALÉM DE ALGUNS PROBLEMAS NOVOS"

Sabemos que isso simplesmente não pode ser reduzido a um problema de gerenciamento. Muitas organizações passam por dificuldades ao realizar a SOA adequadamente, não por causa de problemas técnicos, mas devido a uma falta de governança apropriada. Como você financia um caminho duplo? Como você gerencia o caminho duplo de forma que os objetivos de serviços sejam consistentes com os objetivos de negócio, para que a SOA possa ser gerenciada de forma eficiente? E o que acontece quando não são consistentes umas com as outras?

Precisamos nos afastar da idéia de um comitê sentado em volta de uma mesa ouvindo os relatórios de progresso e os registros de problemas dos gerentes de projeto. Precisamos nos ater à idéia de que a direção envolve o comprometimento com vários acionistas, e esse comprometimento está incluso em uma noção de governança que é crítica para resolver problemas de valor, valor para quem e valor para que finalidade. Essas são questões essencialmente éticas (no sentido mais amplo da palavra ética – a ciência do valor). Assim, enquanto o gerenciamento tem a ver com realizar as coisas, a governança tem a ver com garantir que as coisas certas sejam feitas da forma certa.

Em uma organização hierárquica de TI na qual o "certo" é definido no topo, essa distinção talvez não pareça importar muito. No entanto, caso estejamos pensando no desenvolvimento de um caminho duplo, quando houver tensões necessárias entre as metas comerciais e as necessidades de serviço, sem falar no desenvolvimento federado/distribuído no qual essas tensões são replicadas em várias entidades comerciais, então, aí sim, a distinção importa muito. Tão logo duas atividades paralelas (por exemplo, a criação de serviço e o consumo de serviço) não se reportem verticalmente ao mesmo ponto de gerenciamento, a governança se torna uma questão de negociação entre duas organizações separadas, e não uma simples resolução administrativa em uma única estrutura certa.

Colocando a questão de outra forma, o comprometimento hierárquico depende da transparência vertical – o que pode ser visto é aquilo pelo que você pode se responsabilizar. A transparência vertical não implica transparência horizontal em uma ou mais hierarquias verticalmente transparentes. Sem a transparência horizontal, como os prestadores de serviços podem se sentir responsáveis pelas necessidades de serviços e/ou por outras entidades comerciais?

A governança precisa determinar como os conflitos de interesse entre os acionistas são representados e contidos nos interesses do todo. Em outras palavras, ela precisa criar uma estrutura dentro da qual essa representação seja possível. Se compreendemos que os interesses dos acionistas são expressos como valor a partir de pontos de vista em particular (em uma relação horizontal ou vertical para o que está acontecendo), então, essa questão pode ser formulada em termos de uma estrutura dentro da qual se podem distribuir os benefícios, os custos e os riscos entre esses interesses. Por exemplo, a governança SOA fornece regras básicas para os acordos de interface que precisam ser feitos e aplicados. No mundo real, sabemos que todos os acordos estão sujeitos a serem renegados e renegociados à medida que mudam as exigências e condições. No entanto, que incorre o risco de tais mudanças e quem deve arcar com o custo dessa mudança? Se você decidir que precisa mudar a interface, serei obrigado a responder a essa mudança? E, em caso positivo, com que rapidez e quem pagará a conta?

## Estruturando a transparência

Também há um conflito de interesse entre o presente (adaptação em curto prazo) e o futuro (capacidade de adaptação em longo prazo). Como a capacidade de adaptação será suportada, como se permitirá que as soluções grandes e complexas evoluam (e, além disso, recebam estímulos) e como essa evolução se equilibrará em relação à necessidade de haver ordem e viabilidade de curto prazo?

Essas questões podem ser feitas em dois níveis: primeiro no nível do gerenciamento, em que há uma série de compensações e controles a serem mantidos, e segundo no nível da governança, em que precisamos fazer perguntas sobre a estrutura dentro da qual se podem distribuir os benefícios, os custos e os riscos em uma base contínua.

Em outras palavras, é uma questão de saber como determinamos as responsabilidades e os comprometimentos, e os processos de negociação entre eles. Por fim, é uma questão de saber como estruturar a transparência: determinar quem pode saber o quê, como e quando as coisas estão sendo feitas em relação a quem. A governança reveste-se de autoridade, pois, com comprometimento associado, ela cria responsabilidades sobre o conhecimento e o trabalho mobilizados pela administração (ou, melhor, administrações), exigindo que sejam criadas as formas apropriadas de transparência.

Um dos princípios-chave para o gerenciamento da complexidade em TI e em outros lugares é a separação das preocupações. A separação das preocupações implica atenção seletiva: quem presta atenção a quê. Há uma função importante da governança aqui. A governança não deve se certificar apenas de que a atenção é integral (pelo menos no sentido de que tudo importante está sendo atendido), eficiente (sem atenção dupla desnecessária) e conectada (no sentido de que as preocupações possam ser integradas quando necessário). Ela também deve se certificar de que existam as condições de transparência nas quais tal atenção se faz possível.

**Tabela 1** Comparando a SOA 1.0 e a SOA 2.0

| SOA 1,0 | SOA 2.0 |
|---|---|
| Orientado para o lado da oferta | Colaboração de oferta e procura |
| Processamento direto | Sistemas complexos de sistemas |
| Mente com direcionamento único | Composição colaborativa |
| Reutilização controlada | Reutilização não controlada |
| Endointeroperabilidade (em uma única empresa ou em um sistema colaborador fechado) | Exointeroperabilidade |
| Economia de custo | Experiência aprimorada do usuário |

**Tabela 2** Do desmembramento à interdependência

| Desmembramento | Interdependência |
|---|---|
| "A estratégia de Bush – encher o céu de balas e pão – também é uma aposta, reconhecem os oficiais do Pentágono. A missão humanitária complicará em algum grau o planejamento de guerra. O que os militares chamam de desmembramento – ter certeza de que os aviões de guerra e de ajuda humanitária não vão se confundir – é um ponto muito importante na estratégia do Pentágono. 'Tentar lutar e tentar alimentar ao mesmo tempo é algo novo para nós', afirmou um general das forças aéreas. 'Não temos muita certeza de como tudo vai funcionar'." | "Passamos do desmembramento da capacidade conjunta para a interoperabilidade e depois para a interdependência, quando nos tornamos mais dependentes das capacidades uns dos outros para obter o que precisamos." – Entrevista com o general Shoomaker, CSA, outubro de 2004 |

Criar essas condições de transparência implica uma separação prévia das indiferenças. Separar o que precisa receber a atenção devida do que pode ser ignorado (com segurança) nos leva a uma grande preocupação da governança: a governança do que pode ser ignorado que seja anterior à governança dos conflitos de interesse entre as preocupações. O que qualquer um pode ignorar? Que formas de ignorância são obrigatórias?

Por exemplo, a SOA herda as noções de transparência do trabalho anterior sobre o processamento distribuído aberto. Você pode usar um serviço sem conhecer sua localização; pode usar os dados sem conhecer sua fonte (proveniência). Esse "sem conhecer" é muito útil de algumas formas, mas perigoso de outras. Implica que não há necessidade de haver transparência horizontal.

A SOA envolve o livre acoplamento (e a coordenação horizontal) não apenas entre os artefatos de software, mas também entre as unidades da organização que são responsáveis por esses artefatos. A estrutura da organização geralmente (embora nem sempre) reflete a estrutura do software.

O que acontece com a governança quando não podemos mais confiar na idéia de que alguém mais está cuidando desse risco?

## Interoperabilidade

Às vezes se supõe que a agenda da SOA tenha a ver com "desacoplamento". Utilizam-se os modelos de exigências para promover a decomposição – a identificação de serviços separados que podem ser usados de forma independente de cada um. Esses serviços separados são então projetados para se obter o máximo de reutilização, gerando economias de escala de baixo nível, ao atender a um nível de demanda de qualquer tipo determinado maior do que foi anteriormente possível, e economias de escopo ao atender à demanda de uma variedade maior de contextos.

Há, claramente, alguns sistemas que são excessivamente rígidos e aproveitam o mínimo de folga que se consiga, de tal modo que seus serviços constituintes possam ser usados de forma independente do

**Figura 1** Relacionando o lado da oferta e o lado da procura

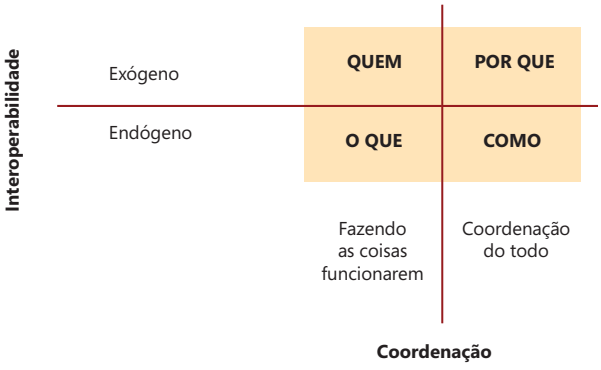
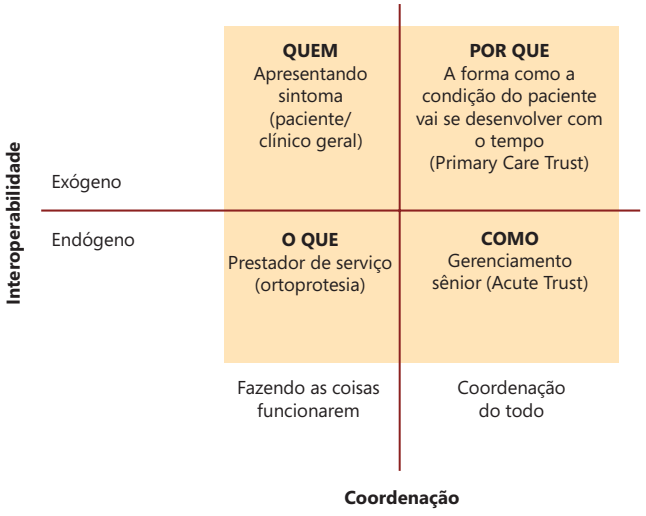



**Figura 2** Relacionando os lados da oferta e da procura para o exemplo da assistência médica




sistema como um todo. Essa rigidez é apenas um lado da história, no entanto. Embora alguns sistemas sejam excessivamente rígidos, há muitos outros que são desesperadamente fragmentados: para se conseguir eficácia ao utilizá-los, os serviços independentes precisam funcionar juntos. Assim, todo o potencial da SOA vem da decomposição e da recomposição.

Um tipo importante de desacoplamento, como praticado pelo setor militar, é conhecido em inglês como "deconfliction" ou "desmembramento". O desmembramento consiste em assumir uma missão inteira e dividi-la em missões menores que podem ser assumidas independentemente umas das outras, o que produz uma cadeia hierárquica de comando na qual a missão de qualquer componente integrante vai depender da forma como a missão se encaixa em um todo maior, definido pelo modo que o comando impõe o desmembramento. O desmembramento se relaciona, em particular, aos efeitos que qualquer missão cria e aos efeitos colaterais desses efeitos em qualquer outra missão. O desmembramento, portanto, exige não apenas a compreensão de como as coisas funcionam (ou seja, o gerenciamento), mas também a compreensão de como os efeitos compostos podem ser obtidos dos efeitos integrantes com o mínimo de conflito entre os componentes integrantes e o máximo de eficiência na utilização de recursos. Assim, o desmembramento tem a ver com o desacoplamento, mas, fundamentalmente, com a forma que esse desacoplamento é feito em relação à missão como um todo. (Consulte a Tabela 2.)

O setor militar leva muito a sério os conflitos entre os componentes integrantes de uma força – o chamado "fogo amigo" (quando matamos nossas próprias tropas por engano) é claramente uma questão de vida ou de morte. Nos nossos termos, o fogo amigo é um exemplo extremo de falha de interoperabilidade. O desmembramento significa organizar as operações de uma forma que minimize o risco em potencial desse tipo de conflito, para que as unidades ou atividades separadas possam ser operadas de forma independente e assíncrona, contribuindo, ao mesmo tempo, para a missão como um todo.

Mas geralmente o desmembramento também envolve uma compensação onerosa. Os recursos precisam ser duplicados, e as operações potencialmente conflitantes são deliberadamente inibidas. As pressões para se conseguir um uso mais eficiente dos recursos forçam uma especialização capaz de garantir economias de escala e escopo, combinadas com missões cada vez mais dinâmicas e políticas, a fim de confrontar qualquer abordagem escolhida para o desmembramento, com níveis cada vez maiores de interdependência.

A resposta a essa questão, à medida que as tecnologias de comunicações e controle se tornaram mais sofisticadas e confiáveis, é aumentar o grau de coordenação possível entre os componentes integrados de uma força, a fim de permitir que unidades e atividades sejam idealizadas de forma mais poderosa, que é o motivo da luta centrada na rede. Em vez de depender de um plano prévio baseado em um desmembramento particular, a rede permite que os comandantes coordenem dinamicamente as relações entre os componentes da força como efeitos compostos necessários para produzir eles mesmos a mudança. É esse uso da rede que possibilita levar o poder ao limite da força quando ele atinge o inimigo.

**"IMPLÍCITA A QUALQUER FORMA DE DESMEMBRAMENTO ESTÁ A ABORDAGEM DE COORDENAR AS ATIVIDADES DESMEMBRADAS POR MEIO DA MANEIRA COMO ELAS INTEROPERAM"**

## Riscos de interoperabilidade

As organizações comerciais e administrativas geralmente testam o desmembramento por meio de uma hierarquia de gerenciamento e por meio de uma estrutura associada de contabilidade dos orçamentos e centros de custos que criam uma transparência vertical consistente com a forma de desmembramento. Sabe-se que esse processo é inflexível e ineficiente, produzindo silos de atividade que são relativamente insensíveis às demandas para organizações diferentes de suas atividades. O poder ao limite (e, discutivelmente, formas avançadas de SOA) não é compatível com as estruturas tradicionais de orçamento e contabilidade dos custos.

O desmembramento nos leva a uma noção muito ampla de interoperabilidade: X e Y são interoperáveis se puderem operar lado a lado sem interferência mútua. Essa operação é o impulsionador por trás das agendas de desacoplamento da SOA, gerando aperfeiçoamentos nas economias de escala e escopo.

Também há uma noção positiva de interoperabilidade: X e Y são interoperáveis se puder haver alguma coordenação ativa entre eles. Essa noção nos força a ir além do desmembramento "per se", e não apenas considerar as pressuposições sobre composição implícitas na forma de desmembramento, mas também pensar em como podem ser explicitadas e dinamizadas pela coordenação. No entanto, trata-se agora de coordenação (horizontal) de rede, e não planejamento hierárquico de cima para baixo, que levanta as mesmas dúvidas de governança que discutimos antes na relação entre o caminho duplo de perseguir metas comerciais e de serviço.

Nosso foco particular encontra-se nos riscos associados à interoperabilidade, e não apenas porque qualquer geometria dada

**Tabela 3** Padrões de relação da governança

| Padrão | Descrição |
|---|---|
| Comparação (QUEM-COMO) | A demanda é definida de tal forma que o contexto do qual surge é ignorado, mas o serviço é coordenado de forma que lhe permita responder àquela demanda em particular. Essa característica corresponde à abordagem da "comparação" para o paciente, quem está procurando a melhor solução oferecida para a demanda dele. Para os fornecedores, essa forma de governança lhes permite minimizar sua exposição aos exo-riscos ao limitar a definição de demanda à qual eles vão responder (a exigência do usuário), embora ainda encarem os riscos de integração dentro da empresa. |
| Custo (QUEM-O QUE) | Não apenas a demanda é definida de forma que o contexto do qual ela surge seja ignorado, como a resposta do serviço também é dividida em procedimentos, e não há necessidade de haver um processo de coordenação explícito. Nessa abordagem de "custo", a natureza das demandas e as respostas a elas se tornaram padronizadas. Agora o risco de integração é minimizado para o fornecedor e para os riscos de tecnologia e engenharia que estão sendo proscritos. |
| Personalização (POR QUE-O QUE) | Uma forma implícita de coordenação de como as coisas funcionam, em geral na forma de um regime orçamentário em particular, restringe a forma como o serviço consegue responder à condição do paciente. Essa característica corresponde à abordagem de "personalização", em que o serviço é padronizado, mas a forma como é prestado de acordo com o contexto do paciente pode ser variada (personalização em massa). Aqui o fornecedor fica novamente exposto aos riscos de integração, à medida que desenvolve mais variabilidade na forma como seus serviços funcionam. |
| Destino (POR QUE-COMO) | Cada uma dessas três formas trata da demanda como algo simétrico a uma forma implícita ou explícita de coordenação endógena. Apenas no quarto caso temos governança assimétrica na qual as formas endógenas e exógenas de coordenação precisam ser alinhadas umas com as outras. Essa característica corresponde à abordagem de "destino", em que o paciente vai para aquele lugar onde pode obter um tratamento exatamente adequado à natureza de sua condição. Também é apenas nesse caso que o fornecedor assume o controle dos riscos de exointeroperabilidade de forma explícita. |

é uma coordenação particular entre seus serviços integrantes, mas também porque são eles que emergem à medida que você se coordena entre os sistemas e organizações. (Estamos acompanhando os recentes desdobramentos sobre o furacão Katrina com considerável interesse, pois algumas das críticas públicas feitas pela FEMA (Federal Emergency Management Agency) expõe claramente as dificuldades de gerenciar alguns dos riscos de interoperabilidade.) Como enxergamos a natureza desses riscos?

A governança aqui envolve o estabelecimento dos termos de referência com base nos quais a administração se encarrega de tornar certas coisas interoperáveis. Os riscos de interoperabilidade podem ser classificados pela gravidade: neste contexto, essa classificação indica até que ponto um risco determinado compromete a forma que desejamos coordenar as coisas.

Implícita a qualquer forma de desmembramento está a abordagem de coordenar as atividades desmembradas pela maneira como elas interoperam. Uma visão simples de interoperabilidade é a que introduz uma forma de ignorância deliberada e seletiva: se você presta atenção em X, não precisa prestar atenção em Y. Por exemplo, se você adota esse padrão aberto, não precisa se preocupar em saber quais dessas plataformas que seguem padrões podem ser usadas. Essa interoperação é uma forma de especialização (ou separação de preocupações) como descrito anteriormente, que possibilita essa ignorância seletiva. Baseia-se em uma pressuposição sobre o que X e Y significam para aqueles que tentam usá-los para produzir um efeito combinado.

Vamos pensar na interoperabilidade das planilhas de gerenciamento em uma grande organização. Cada administrador produz suas próprias planilhas de uma forma idiossincrática para apoiar um conjunto particular de decisões de administração. Embora todos importem alguns dados do banco de dados da empresa, em sua maioria eles acrescentam dados de outros lugares e têm todas as coisas formatadas de forma distinta. Um administrador antigo na empresa, Joe, faz uma apresentação numa reunião do conselho sobre uma importante decisão estratégica, embasando suas recomendações com gráficos elaborados no Microsoft Excel, os quais foram criados a partir de uma complicada planilha desenhada à mão (e totalmente sem referências). Os colegas de Joe acham que é impossível entender a planilha ou importar a análise que ele fez para suas próprias planilhas para análise futura. É bem provável que o sucessor de Joe na empresa crie uma nova planilha, em vez de tentar usar a atual.

Aqui a interoperabilidade falha em dois níveis. Não apenas no nível técnico de compartilhar a planilha como artefato projetado pelo usuário, mas também falha em termos de significado. O artefato é uma expressão de uma estrutura de significado criada por Joe que não é compartilhada pelos seus colegas e sucessores. Joe está tentando coordenar os dados de uma forma que o força a torná-los interoperáveis de formas não familiares (não padronizadas). As muitas dificuldades dos gerentes em colaborar nas decisões estratégicas complexas também refletem o valor em potencial deles ao criar formas de agir.

## Compartilhar o compromisso

O que isso tem a ver com ignorância? Tem a ver com o quanto você precisa conhecer de Joe e de sua experiência de gerenciamento (ou seja, da estrutura de significado dele) para entender a planilha. Quanto mais complexa for a planilha, mais ela se torna quase que uma personificação do próprio Joe e de sua forma de prestar atenção a certos detalhes do gerenciamento. Para usar a planilha, você precisa quase entrar na pele dele, ver as coisas pelos olhos dele. Se Joe é poderoso o suficiente na organização, então ele pode impor sua experiência de administração sobre os colegas e fazê-los usar a planilha, mas em geral isso vai resultar em problemas de interoperabilidade em outro lugar, quando os dados começarem a ser usados de formas novas e não antecipadas. Uma planilha projetada para reutilização precisa assumir algum nível de compreensão compartilhada entre o usuário e o criador.

Para a coordenação entre X e Y, eles precisam claramente conseguir interoperar em um sentido técnico – Joe precisa conseguir me enviar sua planilha, e eu preciso ter os sistemas certos instalados para conseguir abri-la. Mas isso não é o suficiente. Digamos

**Figura 3** O ciclo de governança

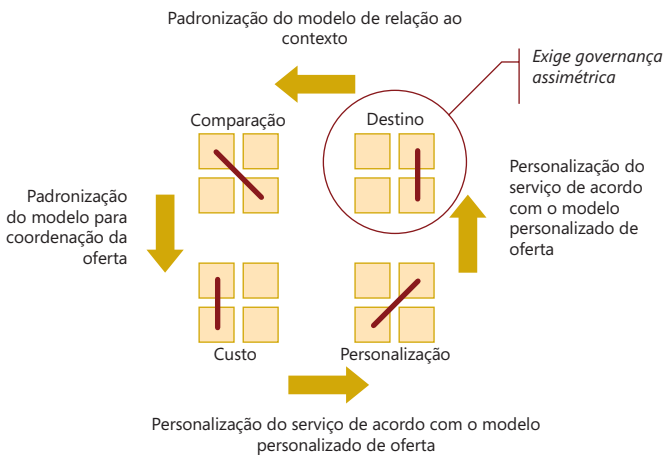
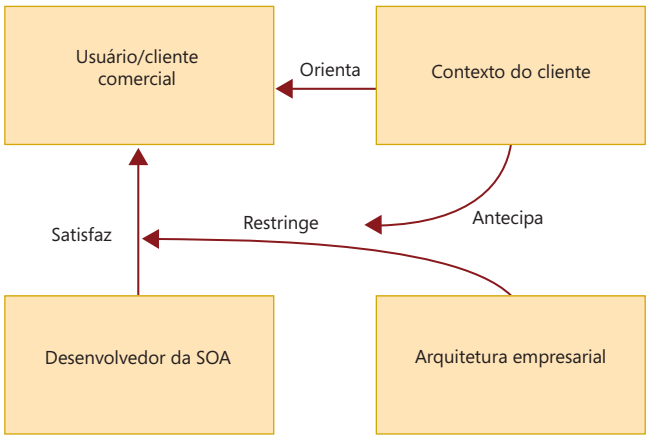


**Figura 4** Estrutura de função




que o P1 usa X e o P2 usa Y. A coordenação precisa considerar os efeitos não apenas de como X e Y interoperam, mas também como P1 e P2 afetam os significados de X e Y. Com a coordenação (ou com sua falta) vem o risco de que a forma como P1 utiliza X e P2 utiliza Y não produza o comportamento composto esperado. Dessa forma, podemos compreender os riscos da coordenação da mesma forma que compreendemos os desafios que a governança de caminho duplo enfrenta. Eles são criados por uma falha no estabelecimento de uma estrutura compartilhada de significado dentro da qual se pode agir.

Com a composição direta (planejamento central, autoridade única de projeto), a questão do significado compartilhado e da ignorância permitida é verticalmente delegada, mas a geometria de negócio resultante precisa ser endógena para as interoperações das atividades sob a hierarquia. Dessa forma, A é decomposto em (ou composto de) B e C; e, se houver riscos associados com a interoperabilidade entre B e C, então esses riscos correspondem à pessoa na hierarquia de projeto que possui A.

Em uma organização, a exigência da transparência vertical pode, portanto, reforçar esse compromisso compartilhado com a coordenação vertical.Em contraste, a composição colaboradora (planejamento no limite entre várias autoridades de projeto) exige que significados compartilhados e ignorâncias permitidas sejam negociados, e que a geometria de negócio resultante tenha elementos exógenos e endógenos que, presumivelmente, não se encontrem todos sob a mesma hierarquia. Dessa forma, a transparência horizontal significa desvendar como todas as peças se juntam em um todo integrado para que o acordo possa ser negociado em termos de como impor uma hierarquia única, ainda que temporária, aos objetivos particulares de colaboração, ou seja, à coordenação horizontal.

Por fim, precisa haver um compromisso compartilhado com uma hierarquia única, independentemente de você estar seguindo ou não um processo de cima para baixo (decomposição analítica e dirigida) ou um processo de baixo para cima (composição sintética e colaboradora), na medida em que possa haver um compromisso compartilhado, no final, com uma hierarquia única. A diferença nas abordagens reside em saber se a hierarquia resultante é ou não estática ou dinâmica. Do ponto de vista de uma empresa que presta serviços a seus clientes, a personalização estática envolverá o consentimento com a hierarquia antes de entrar em uma relação comercial. Em contraste, a personalização dinâmica implica que os próprios processos de concordância com as hierarquias apropriadas precisam fazer parte da relação contínua de negócios.

## Análise de risco

Voltamos à idéia de Alexander do nível no qual podemos estar preservando a estrutura: temos que conseguir decidir quanta geometria precisa ser variável (subdeterminando as maneiras nas quais os usuários de serviço podem personalizar dinamicamente suas utilizações), e quanto dela precisa ser fixa (sobredeterminando as formas de relações comerciais que podem ser aceitas). Ao mesmo tempo em que as formas simétricas de governança podem impor coordenação vertical, as formas assimétricas de governança precisam permitir a coordenação vertical, que levanta novas questões relativas à maneira que a confiança é compartilhada. Na coordenação vertical, ela pode ser garantida pelo contrato com o contexto superior, ao passo que, na coordenação horizontal, precisa ser negociada. Essa questão apresenta novos desafios para a liderança distribuída sob formas assimétricas de governança (consulte Recursos de Huffington et al. e Boxer e Eigen).

Nesses termos, podemos ver que a coordenação vertical é predominante quando o COMO é predominante, ao passo que a coordenação horizontal predomina quando o POR QUE predomina (consulte a Figura 1). Também podemos ver que, na medida em que o COMO continua predominante, estamos externalizando de forma eficaz os riscos exógenos.

Utilizamos métodos de análise organizacional para diferenciar os riscos de endointeroperabilidade (que surgem de falhas dentro da organização) dos três tipos de risco de exointeroperabilidade (nos quais a fonte se encontra fora da organização). Esses riscos de exointeroperabilidade relacionam-se ao que acontece quando os sistemas e serviços de um fornecedor se combinam com os sistemas e serviços de terceiros como parte da solução de um usuário. Dessa forma, o sistema de um fornecedor pode não funcionar conforme esperado no novo contexto, pode não interoperar com outros sistemas conforme esperado e todo o sistema de sistemas pode não interagir com o contexto de utilização do usuário como esperado. Tais resultados podem ser pensados como erros de execução, de planejamento e de intenção dentro do domínio do usuário.

> **"SABER EM QUE PONTO DO CICLO A EMPRESA PRECISA ESTAR É ALGO QUE VAI DEPENDER DA FORMA COMO ESCOLHE EQUILIBRAR RISCOS E BENEFÍCIOS"**

Temos consciência das técnicas analíticas para compreender e gerenciar o risco de endointeroperabilidade. Não temos consciência das técnicas analíticas para compreender e gerenciar o risco de exointeroperabilidade. Essa situação não chega a surpreender em um mundo que define seu negócio como sendo de soluções de push, mas, em um mundo orientado ao serviço, esses riscos começam a se tornar os maiores que existem, na medida em que os fornecedores encontram o pull dos usuários emancipados (consulte Hagel e Brown nos Recursos).

A mudança da impulsão para o empuxo não é apenas uma questão de adotar novas formas de governança e coordenação horizontal. Também exige que o fornecedor adote uma mentalidade de plataforma na qual esta não pertença a ele, mas ao usuário – na melhor das hipóteses, que se forneça uma plataforma na qual os usuários possam resolver seus problemas. A incapacidade geral de gerenciar os riscos de exointeroperabilidade não é um problema muito grande, mas essencial para suportar uma relação de empuxo com o usuário. Muitas das organizações de fornecedores com as quais conversamos ainda negam essa preocupação, mas estamos encontrando mais organizações de usuários que reconhecem a necessidade de seus fornecedores adotarem uma abordagem sistemática para gerenciar os riscos de exointeroperabilidade. O que está em jogo aqui é o velho modelo de compras do estilo "guarda-chuva" que parte do princípio de que as compras do usuário podem ser separadas do contexto de utilização. De fato, os usuários estão buscando uma abordagem colaboradora para gerenciar os riscos presentes quando se fornecem plataformas de serviço.

## Oferta e procura

A característica essencial dessa abordagem de plataforma é que ela gerencia o lado da oferta (ou seja, os níveis de endointeroperabilidade) e o lado da procura (ou seja, os níveis de exointeroperabilidade). Nesses termos, veremos que as quatro posições no ciclo que descrevemos em "The Metropolis and SOA Governance" (The Architecture Journal, Vol. 1, No. 2 – consulte Recursos) constituem quatro padrões diferentes de relações de governança entre os lados da oferta e da procura, e apenas um deles exige uma governança assimétrica. Os outros três utilizam variações de coordenação vertical (consulte a Tabela 3).

Se adotarmos uma abordagem de plataforma, precisaremos conservar as quatro áreas apresentadas na Figura 2 em relação umas às outras, de uma forma que entre em sintonia com o POR QUE – em termos de governança, a coordenação endógena imposta ao serviço pela confiança (o COMO) precisa estar explicitamente alinhada com a forma em particular da coordenação exógena imposta sobre a procura do paciente em decorrência de sua condição (o POR QUE).

O ciclo de governança mostra como dois tipos diferentes de padronização (projetados para reduzir certos tipos de risco) têm o efeito de girar uma empresa ou sistema longe da governança assimétrica, enquanto dois tipos diferentes de personalização (que reintroduz certos tipos de risco) podem fornecer à empresa ou ao sistema acesso aos benefícios em potencial de se comprometer com a assimetria (consulte a Figura 3).

Onde os benefícios se correspondem aos riscos em cada caso? Temos que conseguir entender como essas diferentes formas de governança estão presentes ou são excluídas em uma organização de fornecimento à medida que muda sua relação com a procura, em resposta à alteração das circunstâncias de procura e da concorrência. Na Figura 3, as quatro formas aparecem organizadas, para mostrar o ciclo discutido no nosso artigo "The Metropolis and SOA Governance" (consulte Recursos). Esse ciclo deixa claras as transições entre cada forma; duas delas exigem padronização e duas personalização:

- A mudança do destino para o produto envolve a redução da exposição aos riscos de integração ao se externalizar os riscos exógenos.
- A mudança do produto para o custo envolve a redução da exposição aos riscos de tecnologia e engenharia ao se padronizar o modelo comercial.
- A mudança do custo para a personalização envolve o aumento da exposição aos riscos de integração novamente, mas apenas os endógenos.
- Apenas a mudança da personalização ao destino apresenta à empresa os riscos de exointeroperabilidade.

Saber em que ponto do ciclo a empresa precisa estar é algo que vai depender da forma como escolhe equilibrar riscos e benefícios. Como argumentam Prahalad e Ramaswamy em "The Future of Competition" (consulte Recursos), à medida que a procura se torna cada vez mais assimétrica, também se torna cada vez mais importante para as empresas desenvolver modelos dimensionáveis que também podem funcionar no quadrante de destino. A partir da compreensão de todo o ciclo mostrado na Figura 3, pode-se decidir que formas de mudança precisam capturar a procura, e, portanto, que novas formas de risco precisam ser atenuadas na busca de benefícios e recompensas. Embora se saiba muito sobre como gerenciar três formas de governança nesse ciclo, as dificuldades de conseguir fazer a transição pela quarta forma representa um grande obstáculo ao crescimento contínuo de uma empresa quando confrontada com a assimetria crescente da procura e da demanda.

## No workshop

Voltemo-nos agora a um exemplo de caso da ortoprotesia. Dentro da Acute Trust, foi montada uma clínica para prestar seus próprios serviços de ortoprotesia (comparação), permitindo que a demanda fosse padronizada apenas para essas formas de demanda que surgiam com os consultores. Com o tempo, o próprio serviço e seus orçamentos foram padronizados para se alinhar a essas formas de demanda (custo). Como resultado, os clínicos gerais tinham que se referir aos pacientes pelos consultores, mesmo quando não havia necessidade real de ver o consultor, apenas para obter acesso ao serviço. Como conseqüência,

> "A ASSIMETRIA DA GOVERNANÇA EXPLICA MUITAS DAS DIFICULDADES QUE AS PESSOAS VÊM ENFRENTANDO COM A SOA"

números limitados de encaminhamentos receberam permissão direta para usar o serviço, quando o paciente era conhecido da fundação devido a consultas anteriores (personalização). No entanto, a Primary Care Trust era responsável por todos os pacientes na captação da Acute Trust, e muitos deles não estavam recebendo o serviço necessário. O desafio, portanto, era permitir que o serviço fornecesse suporte diretamente a essas necessidades (destino). A Acute Trust resistiu a esse suporte devido à necessidade de financiar tal serviço de outra forma; e era difícil para a Primary Care Trust começar porque não tinha os meios apropriados para gerenciar o equilíbrio dos custos e dos riscos de tal serviço. Em outras palavras, novas formas de governança assimétrica precisam ser desenvolvidas.

Para trabalhar com essas questões em uma organização de fornecimento, desenvolvemos um processo de workshop que diferencia quatro equipes: azul, branca, vermelha e preta. O processo foi projetado para desempacotar e articular as formas diferentes de risco de interoperabilidade associado a cada um dos quadrantes. O processo de

**Tabela 4 Quatro funções do workshop**

| Equipe | Estilo da equipe | Foco da equipe | Interoperação | Risco | Assimetria |
|---|---|---|---|---|---|
| Azul (O QUE) | Fazemos o negócio | Os recursos que o nosso lado quer usar | Endógeno | I *Comportamento* de tecnologia | Primeiro |
| | | O que o nosso lado consegue fazer | | II *Engenharia* de resultados | |
| Branco (COMO) | Somos os *juízes* do ... | ... que nos *interessa* fazer | | III *Integração* da empresa | Segundo |
| Vermelho (QUEM) | Temos a demanda | O que o lado da procura consegue exigir de nós | Exógeno | I *Execução* da solução | |
| | | A forma como o lado da procura utiliza o que exige | | II *Planejamento* das demandas | Terceiro |
| Preto (POR QUE) | Antecipamos ... | ... quais são os possíveis cenários de acordo com os quais as demandas da equipe vermelha podem surgir | | III *Intenções* no contexto de uso | |

workshop baseia-se deliberadamente em uma metáfora militar, mas foi adaptado para utilização por organizações comerciais e civis, além de instituições militares.

Os objetivos típicos do workshop são aprender coletivamente como essas funções funcionam em relação umas às outras nesta organização específica e neste momento; descobrir até que ponto essa organização carece de capacidade em respeito a essas funções e começar a desenvolver essa capacidade; e obter um panorama da organização atual da demanda que essa organização enfrenta.

As possíveis utilizações desse processo de workshop fornecem uma forma de se abordar uma série de problemas que se relacionam ao impacto das formas assimétricas de demanda, que incluem: estratégia comercial, reprojeto organizacional, projeto e governança da SOA, análise de segurança e governança (consulte a Figura 4).

No nosso exemplo, a equipe azul era o serviço, a equipe branca era a Acute Trust, a vermelha era o paciente e seu clínico geral, e a preta era a Primary Care Trust, agindo de acordo com os interesses dos pacientes em sua captação. A Tabela 4 mostra como essas equipes se relacionam a níveis e assimetrias diferentes. O workshop funciona desta forma:

- Facilita o reconhecimento da parte desempenhada por cada equipe, e as formas em particular de risco que cada uma enfrentava.
- Compreende as conseqüências da presença/ausência de certas cores para o processo de governança.
- Facilita as conversas entre as quatro cores na hora de compreender o sistema como um todo e suas interdependências no gerenciamento dos riscos.
- Usa essa compreensão para desenvolver estratégias a fim de gerenciar os riscos e concordar com a base sobre a qual a equipe branca deve determinar o que há em seus interesses.

## Equipe de trabalho

O foco do workshop encontra-se, portanto, nos problemas em particular que a equipe branca enfrenta na forma como exerce a governança:

- A branca restringe a forma que a azul se comporta em relação aos interesses da branca. Em condições de demanda assimétrica, a branca terá que preferir subdeterminar a azul e permitir a liderança distribuída da azul na forma como esta responde à vermelha em seus contextos particulares da preta.
- Dessa forma, a branca precisa entender a preta para conceder a subdeterminação apropriada à azul, a fim de permitir que a azul satisfaça a vermelha.
- Nesses termos, a governança simétrica da branca torna-se assimétrica, pois dá conta explicitamente das conseqüências da preta em como a azul responde à vermelha.

Nesse caso, o desafio era fazer com que a governança assimétrica da relação entre o serviço e os pacientes na captação da Primary Care Trust fosse exeqüível, o que, por sua vez, significava criar transparência horizontal: era preciso que a clínica fornecesse um relatório da forma que sintonizava o tratamento que realizava com um paciente em particular com a natureza em particular da condição e dos resultados daquele paciente. Assim, a Primary Care Trust estava comprando tratamentos de pacientes, e não números fixos de eventos de tratamento, que envolviam a criação de uma plataforma que poderia suportar essa relação e a mudança dos protocolos de compras, para refletir o eixo alterado da contabilidade. No artigo "The Metropolis and SOA Governance" (consulte Recursos), destacamos o papel desempenhado por essa plataforma em suporte a uma forma diferente de governança. No próximo artigo, vamos analisar os desafios analíticos envolvidos no projeto de uma plataforma como essa, para que ela esteja adequadamente em sintonia com uma forma assimétrica de governança.

A assimetria da governança explica muitas das dificuldades que as pessoas vêm enfrentando com a SOA. Muitas organizações terão equipes de curto prazo em número suficiente para fazer isso de qualquer forma, sem entrar nessa área, mas vamos começar a enxergar as organizações que desejam incorporar esses desafios, e vamos adorar ajudá-las.

Até os anos finais do século XX, poder-se-ia presumir que a grande maioria do tempo gasto por uma empresa seria nas posições simétricas do ciclo. Apenas a geração de novas proposições comerciais precisava ocorrer no quadrante de destino. O desafio apresentado pelo século XXI é que essas proporções estão sendo invertidas. Assim, embora as posições simétricas continuem importantes para colher o valor dos componentes nas plataformas dos usuários, o maior valor será criado, cada vez mais, na parte assimétrica do ciclo. Desenvolver uma capacidade assimétrica é, portanto, de grande importância para as empresas que competem cada vez mais em um mundo horizontal (consulte Friedman em Recursos), no qual as atividades componentes são terceirizadas sob as pressões da globalização, da digitalização e da intensificação da concorrência do lado da oferta. •

### Recursos

Boxer Research Limited
www.asymmetricdesign.com

"From Push to Pull: Emerging Models for Mobilizing Resources", John Hagel and John Seely Brown (October 2005)

"Metropolis and SOA Governance", The Architecture Journal 5, Richard Veryard and Philip Boxer, Vol. 1, No. 2, (Microsoft Corporation, 2005)

"Special Issue on SOA Governance", CBDI Journal (November 2005)

"Taking power to the edge of the organisation: role as praxis", Philip Boxer and Carole Eigen, ISPSO 2005 Symposium, (2005)

*The Future of Competition: Co-Creating Unique Value with Customers, C.K.* Prahalad and Vebkat Ramaswamy (Harvard Business School Press, 2004)

The Nature of Order, Christopher Alexander (Center for Environmental Structure, 2003)

The World Is Flat: A Brief History of the Twenty-First Century, Thomas L. Friedman (Farrar, Straus and Giroux, 2005)

"What is the emotional cost of distributed leadership?" Working Below the Surface: The Emotional Life of Contemporary Organizations, Clare Huffington et al., Tavistock Clinic Series, (H. Karnac Ltd., 2004)

### Sobre os autores

**Philip Boxer** é consultor de estratégia no Reino Unido.

**Richard Veryard** é escritor, consultor administrativo e analista de tecnologia em Londres, no Reino Unido.

# THE ARCHITECTURE JOURNAL™

Input for Better Outcomes

Come visit the Journal's new home at **www.ArchitectureJournal.net** live this December. The new site contains a full library of articles from previous Journal issues in addition to upcoming highlights of our next issue. Browse the content today and post comments and letters directly to the editor!


**Now live at www.ArchitectureJournal.net!**

Microsoft®

ARC

# Aplique os mapas de tópico aos aplicativos

por Kal Ahmed e Graham Moore


## Resumo

Os mapas de tópico fornecem um metamodelo extremamente simples, embora muito poderoso, para a representação dos modelos de conhecimento. Em "An Introduction to Topic Maps" (The Architecture Journal, No. 5, 2005), apresentamos como uma combinação dos mapas de tópico e dos padrões de projeto de arquitetura permite que os desenvolvedores criem componentes reutilizáveis para os aplicativos. Agora vamos dar uma olhada em algumas das áreas atuais e em potencial de aplicação dos mapas de tópicos. Como os mapas de tópicos são utilizados principalmente quando estão integrados a outros sistemas, também vamos discutir o acesso às informações do mapa de tópico usando uma variedade de arquiteturas de serviços da Web, desde uma interação cliente/servidor, estilo RPC tradicional até modelos de distribuição nos quais se podem utilizar tanto um modelo push quanto um modelo pull, a fim de intercambiar as atualizações para um mapa de tópico.


O objetivo principal dos mapas de tópicos é permitir a expressão de um modelo de conhecimento de domínio e possibilitar que esse modelo de conhecimento se conecte aos recursos relacionados. Dentro dessa ampla consideração, podemos identificar várias aplicações comuns para os mapas de tópicos em uma empresa.

De certa forma, a maioria das organizações são agora editoras de recursos. Para algumas, publicar informações é o negócio principal, e para a maioria das outras organizações as informações que publicam fazem parte da comunicação que estabelecem com os clientes e parceiros. As editoras de informações enfrentam uma série de desafios que podem ser resolvidos com os mapas de tópicos.

Em um corpus grande, o mecanismo de pesquisa é geralmente a única forma para os novatos encontrarem o que estão procurando. Tradicionalmente, a pesquisa é determinada pelas palavras-chaves do contexto ou pela indexação de todo o texto. Os mapas de tópicos oferecem a alternativa de indexar e pesquisar com nomes de tópicos, e depois de usar as ocorrências de tópicos para apresentar links para todo o conteúdo relacionado aos tópicos encontrados pela pesquisa. Cada tópico em um mapa representa um conceito único, mas a ele se podem atribuir vários nomes, permitindo que o mapa de tópico armazene nomes científicos e de utilização comum, erros comuns de ortografia ou traduções para os nomes conceituais. Além disso, um mapa de tópicos também pode armazenar relações semânticas entre os termos que podem informar uma pesquisa de várias formas:

1. Essas informações semânticas poderiam ser usadas para fornecer a um usuário sugestões alternativas dos termos de pesquisa ou mesmo expandir de forma transparente o termo de pesquisa inserido. Por exemplo, com as associações apropriadas, um mapa de tópico poderia expandir um termo de pesquisa como "georgiano" para "século XVII E Inglaterra".
2. Seria possível utilizar a digitação de tópicos ou as associações de tópico para tirar a ambigüidade de conceitos distintos com o mesmo termo (homonímia) com base no contexto no mapa de tópicos.
3. Depois de executar uma pesquisa que apresente vários recursos, esses recursos poderiam então ser agrupados de acordo com sua classificação no mapa de tópicos, permitindo que os usuários vejam com mais clareza as diferentes formas por meio das quais a expressão da pesquisa pode ser satisfeita.

### Vantagens para os editores

Uma vantagem diferente para a otimização da pesquisa é que um mecanismo de pesquisa que só investigue um mapa de tópico é significativamente mais fácil de se ajustar. Por exemplo, se uma loja de varejo do setor eletrônico percebe que um novo termo, "PVR", se tornou comum entre os clientes que buscam gravadores de vídeo em disco rígido, o mapa de tópico que controla a pesquisa da loja poderia ser modificado para acrescentar o termo "PVR" ao termo "gravadores de vídeo em disco rígido". O conteúdo vinculado a esse tópico nem precisaria ser modificado; o termo de pesquisa "PVR" buscaria agora o tópico "gravadores de vídeo em disco rígido" e resultaria nos recursos relacionados que estão sendo mostrados.

**"A PRINCIPAL DECISÃO ARQUITETURAL NA IMPLEMENTAÇÃO DOS MAPAS DE TÓPICOS COMO PARTE DE UMA SOLUÇÃO DE PUBLICAÇÃO É COMO O SISTEMA DE MAPA DE TÓPICOS É INTEGRADO AO SISTEMA DE GERENCIAMENTO DE CONTEÚDO"**

*Gerenciamento de links.* Uma das principais maneiras de manter um usuário em um site é apresentar links relacionados que o usuário pode seguir para encontrar mais informações sobre um assunto. Manter esses links manualmente é algo que apresenta propensão ao erro e exige uma troca entre atualizar constantemente os links ou aceitar que o conteúdo mais antigo terá links igualmente antigos ao conteúdo relacionado. A natureza dos mapas de tópicos, como índice de recursos, permite que esses tipos de link sejam gerenciados quase que automaticamente.

Há muitas abordagens diferentes para apresentar links relacionados usando um mapa de tópicos, mas, na essência, todos consistem em sair

do recurso e ir até o(s) ponto(s) no mapa de tópico em que esse recurso recebe referência, e depois cruzar o mapa de tópicos de alguma forma e extrair a lista de recursos que recebem referência a partir do final do cruzamento. Gerenciar links relacionados dessa maneira dinâmica significa que os links relacionados estão sempre atualizados, que a lista de links relacionados pode ser gerada com base na indexação do conteúdo anterior, e que a lógica para extrair os links relacionados pode ser modificada sem a necessidade de modificar os recursos ou a sua indexação.

*Suporte à múltiplas rotas em direção ao conteúdo*. Como foi discutido em nosso artigo anterior, os mapas de tópicos podem ser usados para modelar muitas das ferramentas tradicionais para indexar e encontrar conteúdo, tais como a classificação hierárquica e facetada. Na verdade, um único mapa de tópicos pode conter muitos índices desses. A utilização criativa de diversos índices pode permitir que um usuário encontre conteúdo de muitos pontos de entrada diferentes. Por exemplo, em um site que publica análises financeiras, o usuário pode encontrar um relatório em particular entrando em setores de mercado, indo até a empresa e depois até o relatório, ou por região geográfica até o país relacionado à história, ou até cruzando tópicos personalizados que representam seu próprio portfólio ou interesses.

*Atendendo públicos variados.* Os mapas de tópicos fornecem uma grande flexibilidade àquelas organizações que precisam fornecer acesso a níveis diferentes de usuários. Na primeira instância, os conceitos podem receber nomes conhecidos apenas por determinados públicos. Por exemplo, em um site que fornece informações sobre drogas, o mapa de tópicos poderia fornecer o nome clínico de uma droga para os médicos e o nome comercial para os pacientes, como nomes diferentes no mesmo tópico. Indo mais longe, o mapa pode conter modelos completos e simplificados de domínios que são combinados e que ocasionalmente podem se sobrepor. O principal recurso dos mapas de tópicos para dar conta dessa exigência é a utilização do escopo para especificar o contexto no qual uma dada associação entre tópicos deva ser apresentada ao usuário.

## Arquiteturas de aplicação

A principal decisão arquitetural na implementação dos mapas de tópicos como parte de uma solução de publicação é como o sistema de mapa de tópicos é integrado ao sistema de gerenciamento de conteúdo. A integração precisa ser considerada a partir de dois aspectos: integrar a indexação de conteúdo e a criação de conteúdo e publicar as informações do mapa de tópicos com conteúdo.

Embora a criação de um modelo de conhecimento de domínio possa estar no escopo de um bibliotecário ou no pequeno conjunto de especialistas de domínio, a solução de publicação deve permitir a classificação de conteúdo em relação ao modelo de domínio a ser realizado por aqueles responsáveis pelo conteúdo ou por autores ou editores de conteúdo. Idealmente, a classificação e a indexação do conteúdo em relação ao mapa de tópico devem se tornar uma parte necessária do ciclo de vida de criação/aprovação do conteúdo, com a classificação revisada como parte do processo editorial. Uma aplicação bem-projetada também fará uso de padrões como aqueles que discutimos em "An Introduction to Topic Maps" (consulte Recursos).

Os padrões permitem que um bibliotecário faça mudanças no modelo de domínio que refletem as mudanças na estrutura ou no foco do conteúdo sem exigir nenhuma mudança de fluxo no gerenciamento ou no código de apresentação. Por exemplo, os padrões para os esquemas de classificação hierárquica e facetada permitem que novas hierarquias e novas facetas de classificação sejam introduzidas no mapa de tópicos e sejam reconhecidas e exibidas automaticamente pela camada de apresentação.

A Figura 1 mostra uma arquitetura de blocos simplificada para um site que utiliza um mapa de tópicos. O mapa de tópicos é gerenciado por um componente do mecanismo do mapa de tópico, que é preenchido pelo arquiteto de informações e pelo criador de conteúdo.

**Figura 1** Exemplo de arquitetura para um site orientado a mapas de tópicos

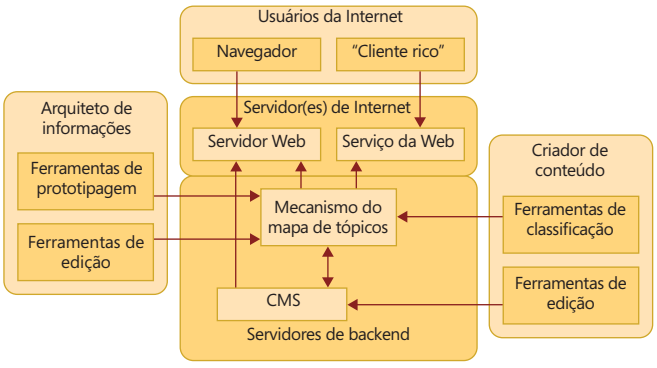
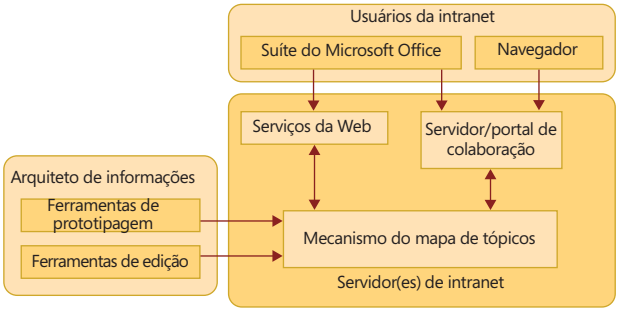



**Figura 2** Exemplo de arquitetura para mapas de tópicos na intranet



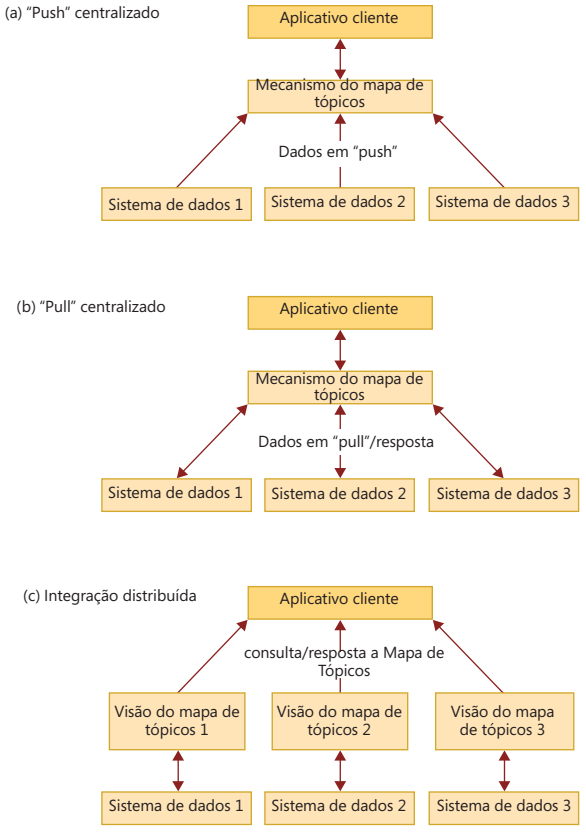

Ao publicar o conteúdo, há duas abordagens possíveis para a integração das informações a partir do mapa de tópicos. Na primeira abordagem, o CMS (Sistema de Gerenciamento de Conteúdo) fornece a estrutura do site e uma ou mais regiões na página nas quais é possível acrescentar as informações do mapa de tópicos. Na segunda abordagem, usa-se uma parte do modelo de domínio do mapa para orientar o site.

A primeira abordagem é a mais apropriada com os CMSs, que têm fortes recursos de gerenciamento do site ou que fazem uso da estrutura do site para implementar controles de acesso ou outros recursos. A segunda abordagem, não obstante, pode ser usada para criar uma estrutura flexível de site que pode ser modificada com mais facilidade para considerar as mudanças no modo como os índices de conteúdo são organizados. Além de estruturar o conteúdo no CMS, o mecanismo de mapa de tópicos pode funcionar como índice útil de todo o conteúdo disponível no site. Esse índice pode ser disponibilizado por meio de uma interface de serviços da Web aos clientes ricos, como um leitor de RSS ou o Serviço de Pesquisa no Microsoft Office (consulte a Figura 1).

Os mapas de tópicos podem suportar um aplicativo corporativo de gestão de conhecimento, principalmente ao fornecer um repositório para capturar o modelo de conhecimento de domínio. O metamodelo flexível fornecido pelos mapas de tópicos permite que novos tipos de conceito e novos tipos de relacionamentos sejam introduzidos no modelo de conhecimento com o mínimo de esforço, possibilitando que o modelo de conhecimento mantenha o mesmo ritmo das mudanças na empresa.

Além de manter o modelo de conhecimento para o domínio, o mapa de tópicos também pode ser usado para indexar os recursos na intranet. Nesse sentido, o mapa de tópicos fornece vantagens semelhantes àquelas descritas para a publicação geral de recursos, mas, com o desenvolvimento de sistemas de colaboração, como o SharePoint, a extensão do que está disponível pela intranet está aumentando rapidamente. Esses sistemas fazem com que fique mais fácil para

**Figura 3** Três arquiteturas de integração das informações empresariais




os usuários criar conteúdo da intranet e compartilhar conteúdos relacionados a um projeto. Mas esse compartilhamento pode levar rapidamente a uma intranet sobrecarregada de dados na qual fica difícil encontrar as informações relevantes e com a qual é quase impossível um novato se familiarizar. Um índice do conteúdo de intranet baseado nos mapas de tópicos não pode simplesmente ajudar os usuários experientes a encontrar o conteúdo relevante independentemente de sua localização; o modelo de domínio de alto nível encorporado pelo mapa de tópicos também pode ser útil para fornecer aos usuários iniciantes uma visão geral das atividades da organização.

## Conexões de conteúdo

Uma parte significativa do problema para a gestão do conhecimento corporativo é que os usuários precisam trabalhar com muitas tarefas que não têm uma correlação direta com algum dado específico. Por exemplo, o usuário pode trabalhar em um caso ou fazer parte de um projeto, ou estar em uma equipe multifuncional, ou ainda participar de uma reunião. Todas essas tarefas podem ter dados relacionados – anotações do caso, documentação do projeto, atas da reunião e assim por diante – mas com freqüência elas mesmas não têm identidade real. Embora se possa usar palavras-chaves para conectar os itens do conteúdo a esses conceitos, uma simples palavra-chave não diz nada aos usuários sobre o que torna o conteúdo relevante para a palavra-chave ou sobre as relações entre as palavras-chaves.

Um mapa de tópicos fornece um modelo para definir como tópicos os itens de não-conteúdo, como pessoas, projetos e locais. Depois que esses tópicos são criados, eles podem então ser conectados aos itens de conteúdo. Os mapas de tópicos fornecem duas maneiras de fazer a conexão com o conteúdo. A primeira é criar um segundo tópico que representa o conteúdo e depois usar uma associação. A segunda é usar uma ocorrência que aponte diretamente para os itens do conteúdo. O modelo também pode ser usado para motivar relações importantes entre os itens de não-conteúdo que estão fora do conteúdo no mapa de tópicos, a fim de fornecer uma visão geral das funções da organização.

Por exemplo, um projeto pode ter muitos participantes e pode ser levado adiante para um cliente em particular. Transformando os participantes e o cliente em tópicos no modelo de domínio, seria possível então localizar rapidamente outros projetos para o mesmo cliente, listar os produtos que foram licenciados ao cliente ou então encontrar os outros compromissos que um membro da equipe assumiu no projeto.

Pode-se usar o mapa de tópicos como uma base de conhecimento em si mesma. Uma aplicação simples seria um portal da Web com uma interface que permite aos usuários criar seus próprios tópicos e associações e navegar até os tópicos e associações criados por outras pessoas. Os usuários também poderiam adicionar links aos recursos internos ou externos como ocorrências, ou a interface poderia permitir que os usuários criassem ou carregassem conteúdo ao próprio portal. Na verdade, trata-se de recursos comuns de muitas aplicações genéricas de edição do mapa de tópicos. No entanto, o maior benefício surge quando o mapa de tópicos é integrado a um sistema de colaboração que pode fornecer gerenciamento de conteúdo e de eventos, fóruns de discussão e outros recursos. Como ocorre com a integração para a publicação de recursos, uma implementação bem-sucedida exige que a funcionalidade do mapa consiga fazer parte da interação normal com o sistema de colaboração.

"TENDO CRIADO TÓPICOS E CATEGORIZADO O CONTEÚDO, O PRIMEIRO ASPECTO PARA A UTILIZAÇÃO DOS MAPAS DE TÓPICOS EM UM CENÁRIO CORPORATIVO DE GESTÃO DO CONHECIMENTO É O ACESSO ÀS INFORMAÇÕES DO MAPA DE TÓPICOS"

Categorizar o conteúdo é algo que sempre será visto pelos usuários como um fardo adicional. Pode-se reduzir esse fardo usando uma classificação orientada a esquemas, para minimizar as decisões que os usuários precisam tomar sobre a classificação dos itens, e utilizando a topologia do sistema de colaboração (por exemplo, todos os documentos colocados na pasta "Projeto X" são marcados automaticamente como relacionados ao projeto "Projeto X"). No outro lado da equação, não é necessária muita classificação para gerar rapidamente vantagens para os usuários ao unir conteúdos distintos, e apenas uma única pessoa trabalhando para categorizar o conteúdo produzido pelo projeto ou pelo departamento pode resultar em uma melhor base de conhecimento para todos os usuários. A natureza inerentemente flexível da ontologia baseada em mapas de tópicos faz com que fique mais fácil começar pequeno, se focar na ontologia central necessária para resolver as exigências apenas daquele projeto ou departamento e depois ir evoluindo, à medida que o projeto mostra um retorno sobre o investimento.

## Fornecendo o conhecimento

Tendo criado tópicos e categorizado o conteúdo, o primeiro aspecto para a utilização dos mapas de tópicos em um cenário corporativo de gestão do conhecimento é o acesso às informações do mapa de tópicos. Esse aspecto é a área em que um mapa de tópico pode realmente se destacar. O mapa de tópicos pode fornecer um modelo estruturado e de alto nível do domínio que é facilmente transmitido por uma interface de serviços da Web, dando um enorme potencial para a integração nos aplicativos desktop. Por exemplo, recentemente desenvolvemos um serviço da Web que implementa uma interface

do Serviço de Pesquisa do Microsoft Office para fornecer ao usuário a capacidade de pesquisar e examinar um mapa dentro do Internet Explorer e dos aplicativos do Office.

A Figura 2 mostra uma possível arquitetura para um aplicativo de base de conhecimento desse tipo. O arquiteto de informações estabelece os esquemas básicos de taxonomia e classificação para a base de conhecimentos. Os usuários então trabalham com o servidor de colaboração e preenchem e ampliam essa taxonomia por meio das interfaces do navegador ou do "cliente rico".

Utilizando as Smart Tags e interfaces mais detalhadas de serviços, como aquelas descritas anteriormente, estão disponíveis possibilidades semelhantes de integração. Essas integrações permitem que as informações em um mapa de tópicos sejam fornecidas diretamente ao aplicativo no qual se mostram mais úteis. Além disso, o modelo de domínio para o gerenciamento interno de conhecimento também pode ser usado como base para integrar os dados de outros sistemas empresariais, como vamos descrever brevemente.

Os mapas de tópicos são freqüentemente usados como complemento dos CMSs, o que é algo que não surpreende, dados os recursos apresentados em termos de organização de conteúdo, indexação e pesquisa. No entanto, os mapas de tópicos também podem ser usados como base para integrar todos os tipos de fontes de dados.

A chave para a integração das informações empresariais, utilizando os mapas de tópicos, é definir uma ontologia principal e depois mapear os dados de cada sistema de dados nessa ontologia. Por exemplo, a ontologia poderia conter os conceitos de um Cliente e de um Pedido, mas é o sistema de CRM (Gerenciamento de Relacionamento com o Cliente) que contém as informações sobre as ligações do cliente para o suporte técnico e sobre o sistema de controle de pedidos que contém as informações sobre o status dos pedidos do cliente. Uma abordagem para integrar esses sistemas com base nos mapas de tópicos poderia ser centralizada ou distribuída. Essas abordagens diferentes são mostradas na Figura 3.

Em um sistema centralizado, cada sistema de dados fornece informações sobre as entidades que gerencia para um mapa de tópicos central. As informações podem ser publicadas por um envio do sistema de dados (consulte a parte A na Figura 3) ou atualizadas usando uma requisição pelo aplicativo central de gerenciamento do mapa de tópicos (consulte a parte B na Figura 3). Outros aplicativos podem então se utilizar dessas informações centralizadas como tópicos, associações e ocorrências, protegendo os aplicativos do cliente da necessidade de conhecer as interfaces necessárias para se comunicar com cada sistema externo. Nesses sistemas, a replicação de dados precisa ser tratada com cuidado para garantir que fique claro onde está o mestre para qualquer item de dados.

## Um serviço simples da Web para mapas de tópicos

Embora seja possível os aplicativos gerarem dados de mapa de tópicos, a variedade de aplicativos que podem criar ou consumir essa fonte de dados é limitada. A idéia do serviço da Web é abrir os mapas de tópicos para mais aplicativos, pouco importando se estão publicando ou buscando informações. Além disso, um serviço da Web bem-definido apresenta uma associação valiosa ao aplicativo no qual os serviços de conhecimentointeragem.

O serviço da Web apresentado aqui compreende um pequeno conjunto de métodos genéricos que garantem que ele pode ser usado em uma ampla variedade de soluções de mapa de tópicos. Descrevemos brevemente os métodos do serviço da Web e fornecemos comentários sobre sua aplicabilidade e seu uso pretendido. Essa interface é implementada pelo Topic Map Web Service WSDL, da Networked Planet (consulte Recursos), que também implementa dos métodos adicionais que permitem um acesso mais direto a quaisquer hierarquias contidas no mapa de tópicos.

**GetTopicMapNames()** – Um sistema de mapa de tópicos pode armazenar e gerenciar muitos mapas de tópicos. Essa operação retorna uma lista de nome dosmapas de tópicos atualmente disponíveis. Além disso, é possível fazer com queesse serviço agregue ou aja como intermediário em vários mapas de tópicos distribuídos. Esse método fornece uma maneira de descobrir quais mapas de tópicos estão disponíveis.

**GetTopic(TopicMapName, TopicId)** – Essa operação retorna uma serialização do tópico especificado a partir do mapa de tópicos nomeado. A serialização fornece ao chamador as informaçõessuficientes que ele pode cruzar com os tópicos relacionados.

**GetTopicTypes(TopicMapName)** – Essa operação retorna uma serialização de um subgráfico do mapa de tópicos na qual cada tópico que funciona como tipo de um ou mais tópicos no mapa especificado é totalmente serializado.

**GetTopicBySubjectIdentifier(TopicMapName, Identifier)** – Essa operação retorna um subgráfico do mapa especificado, no qual o tópico com o identificador especificado como seu identificador de assunto é totalmente serializado.

**GetTopicsByType(TopicMapName, TopicTypeId)** – Essa operação permite que os usuários busquem uma lista de todos os tópicos que são instâncias do tipo determinado. O valor de retorno é um subgráfico do mapa de tópicos com cada tópico de instância totalmente serializado.

**DeleteTopics(TopicMapName, TopicId[])** – Essa operação especifica o identificador de tópicos exclusivo de um ou mais tópicos a serem excluídos do mapa de tópicos especificado.

**SaveTopic(TopicMapName, TopicMapFragment)** – Essa operação pode ser usada para atualizar um tópico existente e adicionar novos tópicos. Os dados do mapa de tópicos contidos no parâmetroTopicMapFragment funciona como a versão definitiva de tópicos e associações. Para substituir um tópico ou associação existente, esse constructo no parâmetro TopicMapFragment deve carregar o identificador exclusivo do constructo a ser substituído. Qualquer tópico ou associação no parâmetro TopicMapFragment que não carrega um identificador exclusivo éadicionado como dado novo ao mapa de tópicos.

**Query(QueryName, QueryParams[])** – O serviço da Web não permite a execução de pesquisas arbitrárias nos mapas de tópicos. Em vez disso, o administrador ou desenvolvedor pode configurar qualquer número de consultas nomeadas que são acessíveis por meio da API do serviço da Web. Essa operação invoca a consulta nomeada, transmitindo parâmetros (espaço) usados como substituições de valor direto na seqüência de consulta. A estrutura de XML retornada deste método depende da estrutura da tabela de espaço pela consulta.

A principal característica dessa interface é que ela é quase inteiramente centrada no tópico. Todas as operações estão ativadas e retornam tópicos ou listas de tópicos. Usando as técnicas de serialização descritas aqui, é possível implementar esse serviço da Web usando uma abordagem centrada em documentos com os dados XML, que são facilmente processados com a vinculação de dados XML ou com os conjuntos de ferramentas XPath.

**Figura 4** Arquitetura intermediária de mapas de tópicos

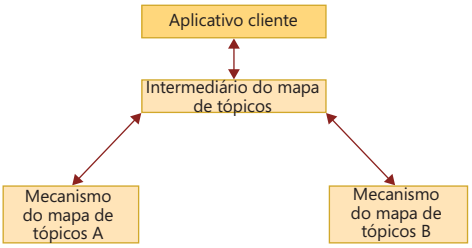
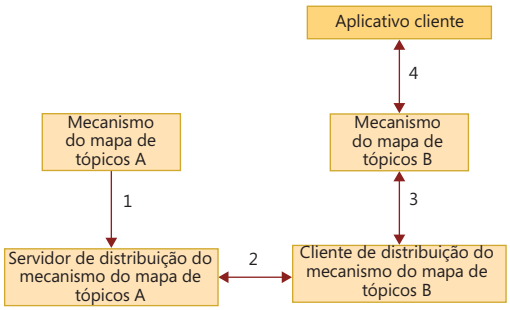



**Figura 5** Exemplo da arquitetura de distribuição para os mapas de tópicos




Em um sistema distribuído, cada sistema de dados tem um componente de integração que expõe uma interface do mapa de tópicos, e os clientes contatam os sistemas por meio dessa interface (consulte a parte C da Figura 3). Novamente, os clientes só precisam entender uma interface, mas, nesse caso, as ligações para a interface do mapa de tópicos poderiam ser convertidas diretamente em investigações e atualizações em relação ao sistema subjacente de informações, o que significa que não há problema na replicação de dados, mas pode levar a uma tarefa de integração mais complicada.

## Identificação e URIs

Em um sistema centralizado ou distribuído, precisa haver uma forte ênfase na identificação das entidades que cada sistema gerencia. É uma questão simples mapear os identificadores únicos à entidade, como o número de conta do cliente ou o número de controle do pedido, em um URI (Identificador de Recursos Uniforme) para o tópico que representa aquela conta ou aquele pedido, e, com um pouco de cuidado, os identificadores podem ser construídos de forma que haja um algoritmo comum para converter entre os URIs e os identificadores específicos da entidade.

A vantagem principal de usar um mapa de tópicos para esse tipo de exercício de integração é a flexibilidade que ele permite para a modelagem dos dados integrados. Como o modelo são simplesmente dados em um mapa de tópicos e não um esquema para qualquer sistema subjacente, novos tipos de entidades podem ser apresentados sem a necessidade de alterar quaisquer interfaces de integração existentes. Além disso, representando entidades comerciais importantes como tópicos em uma ontologia corporativa, fica claro realizar e integrar os dados dos sistemas de informações empresariais no sistema de gestão de conhecimento ou mesmo, quando apropriado, na Web.

Todos os aplicativos que usam os mapas de tópicos, inclusive todos aqueles apresentados anteriormente, exigem algum método para acessar as informações do mapa de tópicos. A maioria dos aplicativos também exige um armazenamento persistente para os dados do mapa de tópicos e uma API de processo para acessar e manipular esses dados, mas estes estão além do escopo desta discussão. O que é mais interessante para o arquiteto de soluções é a capacidade de acessar as informações do mapa de tópicos por meio das chamadas de serviços da Web. Os mapas de tópicos têm várias propriedades que os tornam altamente úteis para acessar os serviços da Web. Vamos dar uma olhada neles.

*Endereçamento do tópico.* Aos tópicos é possível atribuir identificadores exclusivos por servidor (ou, se necessário, globalmente exclusivos). A propriedade de endereçamento de tópico nos permite construir aplicativos cliente que podem controlar a proveniência das informações do tópico que eles utilizam. O direcionamento de tópico também nos permite cruzar a associação ou as informações de classificação na representação de um tópico. Por exemplo, uma investigação de um cliente pode recuperar o tópico A com a ocorrência de que é classificada pelo tópico T. Uma segunda investigação do cliente pode então recuperar todas as informações sobre o tópico T. No serviço da Web de amostra descrito no quadro, "Um serviço simples da Web para mapa de tópicos", os tópicos podem ser recuperados por seu identificador exclusivo usando o método GetTopic().

*Endereçamento de conceito.* É possível atribuir aos conceitos que os tópicos representam identificadores separados e exclusivos, permitindo uma investigação em relação a vários servidores para as informações que se relacionam a um conceito específico. O endereçamento de conceito é essencial para suportar a criação e a manutenção distribuídas de mapas de tópicos e a subseqüente agregação das informações nesses mapas. Já que se pode atribuir um conceito (como Pessoa, Lugar, Fred Jones ou Birmingham) a seu próprio URI separado do identificador de sistema do tópico que representa esse conceito, diversos sistemas podem fornecer informações sobre o mesmo conceito e usar o identificador de sistema como a chave usada na agregação.

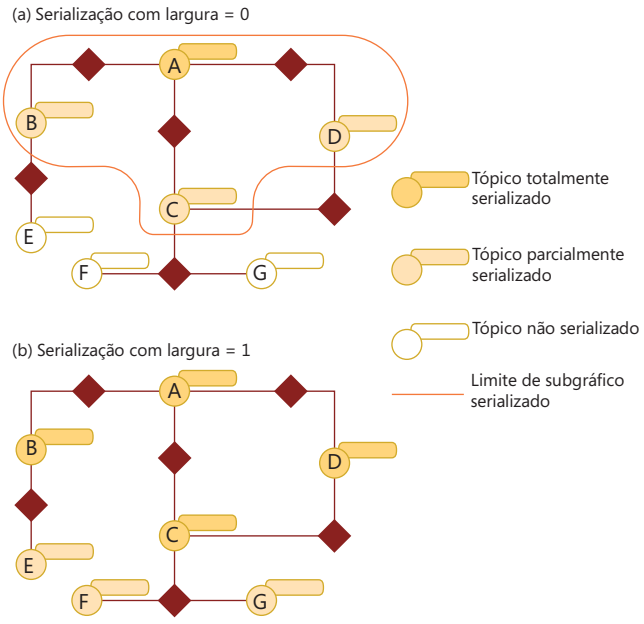
Por exemplo, uma consulta realizada por um cliente pode retornar um tópico T com um identificador de assunto I. O cliente poderia então investigar um segundo mapa de tópicos para qualquer tópico com um identificador de assuntos I para expandir a quantidade de informações conhecidas sobre o conceito representado por aquele identificador. A vantagem do endereçamento de conceito é que o aplicativo cliente não precisa conhecer o identificador específico de sistema do tópico com o identificador de assunto I. No serviço da Web de exemplo, essa forma de direcionamento é fornecida pelo método GetTopicBySubjectIdentifier() (consulte o artigo "Um serviço simples da Web para mapa de tópicos").

**"É POSSÍVEL ATRIBUIR AOS CONCEITOS QUE OS TÓPICOS REPRESENTAM IDENTIFICADORES SEPARADOS E EXCLUSIVOS, PERMITINDO UMA INVESTIGAÇÃO EM RELAÇÃO A VÁRIOS SERVIDORES PARA AS INFORMAÇÕES QUE SE RELACIONAM A UM CONCEITO ESPECÍFICO"**

*Tópicos literais de documentos.* Os tópicos podem ser exibidos facilmente como estruturas de dados que utilizam identificadores de tópicos globalmente exclusivos ou hiperlinks para representar as relações entre eles. Nos termos SOAP, isso significa que podemos criar uma representação simples e literal dos documentos de um tópico. Se o REST (Representational State Transfer) for a sua metodologia preferida de serviço da Web, é possível construir uma representação de tópico como documento com hiperlink enviado em resposta a uma investigação de REST completa. Descrevemos brevemente o algoritmo para produzir essa representação.

*Regras padronizadas de mesclagem.* As regras de mesclagem dos mapas de tópicos podem ser usadas para combinar informações de tópicos recebidas de várias fontes separadas em um único mapa de tópico ativo. Para os clientes de mapas de tópicos, as regras de mesclagem permitem que os clientes usem o endereçamento de

**Figura 6** Serialização do subgráfico do mapa de tópicos




conceito para encontrar todas as informações relacionadas a um conceito e depois combinar essas informações em um único tópico que é apresentado a níveis superiores do aplicativo. Essa mesclagem permite que um cliente agregue informações de várias fontes de mapas de tópicos e depois apresente uma interface que deixe claro que todas essas informações agregadas vieram de uma única fonte. As regras de mesclagem também permitem que se dê um passo a mais no endereçamento de conceito, pois possibilita que um mapa de tópicos declare que dois conceitos são equivalentes simplesmente incluindo o endereço de cada conceito em um único tópico.

Por exemplo, um cliente poderia investigar quaisquer informações relacionadas a um conceito com o identificador I e o mapa de tópicos poderia retornar um tópico T que tem os identificadores I e I' como seus identificadores de assuntos, e isso diz ao cliente que a fonte está reivindicando que o conceito identificado por I seja igual ao conceito identificado por I'. Se o cliente confia na fonte para fazer esse tipo de reivindicação, ele pode então continuar investigando qualquer informação relacionada ao conceito com o identificador I' e mesclar isso com as informações já recebidas para o conceito com o identificador I.

Os mapas de tópicos podem ser usados facilmente com quase qualquer arquitetura de acesso de cliente. Vamos examinar três arquiteturas comuns: cliente/servidor, broker e de syndication.

A mais simples das arquiteturas de acesso do mapa de tópicos, a arquitetura cliente/servidor, utiliza uma interface de serviço da Web que expõe as operações de navegação, consulta e atualização de um mapa de tópicos. (Consulte o artigo, "Um serviço simples da Web para mapa de tópicos", para obter uma descrição de uma interface possível que consiste em apenas oito métodos.) Como os mapas de tópicos podem ser facilmente serializados como XML, não há problema em usar SOAP, REST ou até XML-RPC para implementar uma interface desse tipo.

A arquitetura broker interpõe um ou mais servidores adicionais entre a(s) fonte(s) dos dados do mapa de tópicos e os clientes (consulte a Figura 4). Um intermediário agrega os resultados de vários servidores realizando qualquer mesclagem necessária e respondendo a um cliente como se os dados tivessem vindo de um único mapa de tópicos. A agregação realizada por um intermediário pode variar da simples distribuição de uma operação e da agregação dos resultados até a agregação mais complexa baseada nos identificadores de assuntos devolvidos de várias fontes de mapas de tópicos.

## Arquitetura de syndication

A arquitetura de syndication usa o REST para distribuir as mudanças em um modelo de mapa de tópicos. O servidor que mantém o modelo simplesmente grava as mudanças como documentos de transação que contêm subgráficos serializados de mapas de tópicos. Esses documentos de transação são então utilizados pelos clientes e aplicados ao cache local do modelo. As arquiteturas de syndication funcionam particularmente bem para distribuir os mapas de tópicos de ontologia que são relativamente estáveis ou para distribuir os mapas que indexam o conteúdo que é publicado regularmente (por exemplo, as atualizações de notícias de um site). A serialização XML dos subgráficos do mapa de tópicos indica que essa arquitetura pode fazer uso dos padrões de distribuição como ATOM ou RSS 2.0 para distribuir os dados de transação.

A Figura 5 mostra um exemplo de como o sistema de distribuição pode funcionar para manter sincronizados dois mapas de tópicos gerenciados por mecanismos diferentes:

**"O MAPA DE TÓPICOS PODE FORNECER UM MODELO ESTRUTURADO E DE ALTO NÍVEL DO DOMÍNIO QUE É FACILMENTE TRANSMITIDO POR UMA INTERFACE DE SERVIÇOS DA WEB, DANDO UM ENORME POTENCIAL PARA A INTEGRAÇÃO NOS APLICATIVOS DESKTOP"**

1. Uma mudança feita no mecanismo do mapa de tópicos A é gravada em um documento XML no servidor de syndication. O servidor de syndication faz a alimentação dos documentos recentes da transação disponíveis nos clientes de distribuição.
2. O cliente de syndication verifica a alimentação no servidor de syndication e solicita a(s) transação(ões) que ele precisa aplicar.
3. O cliente de syndication processa os documentos de transação recebidos do servidor de syndication e aplica as mudanças ao mecanismo do mapa de tópicos B.
4. Os clientes interagem com o mapa de tópicos atualizado no mecanismo de mapa de tópicos B, que agora é sincronizado com o último estado conhecido do mapa no mecanismo do mapa de tópicos A.

A etapa 2 também poderia ser realizada por uma pressão das informações de transação distribuídas do servidor de syndication para o cliente; o mecanismo usado dependeria das exigências do aplicativo e das instalações fornecidas pelo mecanismo de syndication utilizado.

Os aplicativos que acessam os mapas de tópicos freqüentemente exigem um conjunto de resultados que consiste em um único tópico (ou uma lista de tópicos) que corresponde à investigação. No entanto, o modelo de mapa de tópicos é essencialmente um modelo de grafo no qual os tópicos são conectados por associações ou relações de classificação; assim, ao retornar um tópico, é necessário que o servidor forneça algum contexto para o cliente. Basicamente, o servidor precisa extrair um subgrafo do grafo do mapa de tópicos.

## Serialização do mapa de tópicos

Na experiência que tivemos, a melhor maneira de tratar da serialização é começar com o conceito de dois tipos de serialização de tópico. Uma serialização completa apresenta todas as informações diretamente conectadas a um tópico: tipos, identificadores, nomes, ocorrências e todas as associações das quais participa. Uma serialização parcial apresenta um conjunto mínimo de informações que podem ser

usadas por um cliente. Determinar exatamente quais informações estão presentes em uma serialização parcial pode variar de uma implementação para outra; para algumas implementações, apenas um identificador exclusivo de tópicos pode ser suficiente, mas outras implementações podem exigir que todos os identificadores estejam presentes em um tópico, além do nome ou da ocorrência do tópico escolhido de acordo com algum algoritmo. O principal guia na hora de determinar o que está presente em um tópico parcialmente serializado é que uma serialização parcial não deve conter nenhuma referência a outros tópicos; assim, os tópicos parcialmente serializados formam as autorizações do subgráfico retornado por uma serialização.

Com essas duas definições, a extração do subgráfico é uma tarefa relativamente direta. Para extrair um pequeno subgráfico centrado em um tópico, realize uma serialização completa daquele tópico. Para cada tópico ao qual o tópico totalmente serializado faz referência, crie uma serialização de fragmento do tópico e substitua a referência a ele pela referência ao fragmento.

Subgráficos maiores podem ser extraídos especificando-se um parâmetro de largura que define o número máximo de associações a serem cruzadas. Cada tópico que pode ser alcançado ao se cruzar uma série de associações até o parâmetro de largura a partir do tópico de início deve ser inteiramente serializado, e todos os outros tópicos referenciados devem ser serializados como fragmentos. A Figura 6 mostra um exemplo da serialização de um subgráfico de mapa de tópicos usando duas larguras diferentes de subgráfico.

A parte A na Figura 6 mostra que a serialização do tópico A é realizada com uma largura de 0, assim apenas o tópico A é totalmente serializado. No entanto, para serializar as associações das quais o tópico A participa, cada um dos tópicos B, C e D deve ser parcialmente serializado. Observe que o tópico C tem uma associação ao tópico D, mas, como o C é apenas parcialmente serializado, essa associação não faz parte do subgráfico serializado mesmo que suas extremidades sejam.

A parte B na Figura 6 mostra a serialização do tópico A com a largura de 1. Nessa serialização, o tópico A e todos os tópicos que podem ser alcançados ao se cruzar uma associação que começa no tópico A (ou seja, tópicos B, C e D) são totalmente serializados. Para realizar a serialização completa do tópico B, o tópico E deve ser parcialmente serializado, e para realizar a serialização completa do tópico C, os tópicos F e G devem ser serializados. O tópico C só é conectado aos tópicos que são totalmente serializados, assim não é necessária nenhuma informação extra para serializar as associações das quais ele participa.

**"NA EXPERIÊNCIA QUE TIVEMOS, A MELHOR MANEIRA DE TRATAR DA SERIALIZAÇÃO É COMEÇAR COM O CONCEITO DE DOIS TIPOS DE SERIALIZAÇÃO DE TÓPICO; A SERIALIZAÇÃO COMPLETA APRESENTA TODAS AS INFORMAÇÕES DIRETAMENTE CONECTADAS A UM TÓPICO"**

Tendo identificado o subgráfico do mapa de tópicos a ser extraído, tudo o que resta é para serializar os dados naquele subgráfico como XML. Embora o padrão de mapas de tópicos defina uma sintaxe de troca em XML, ele é uma sintaxe mais bem ajustada para criação de autoria e intercâmbio de todos os tópicos, não fornecendo nenhuma sintaxe para diferenciar entre os tópicos totalmente e parcialmente serializados. Portanto, é necessário definir um esquema separado para serialização dos subgráficos do mapa de tópicos. Ao projetar nosso próprio esquema (www.networkedplanet.com/2005/01/topicmap/data/TopicMapFragment.xsd), também aproveitamos a oportunidade de simplificar a sintaxe XTM para remover os recursos de conveniência de criação de autoria, resultando em um esquema que pode ser processado mais facilmente pelas ferramentas de vinculação de dados em XML e pelo XSLT/XPath.•

### Recursos

"ISO/IEC 13250 Topic Maps", Second Edition (2002)
www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-edv2.pdf

"ISO/IEC JTC1/SC34 Information Technology –
Document Description and Processing Languages"
www.isotopicmaps.org/sam/sam-xtm/

Networked Planet
Topic Map Web Services
www.networkedplanet.com/technology/webservices/intro.html

Topic Map Web Service WSDL
www.networkedplanet.com/2005/01/webservices/tmservice/TMService.wsdl

"White Paper: Topic Maps in Web Site Architecture",
(Networked Planet Ltd., 2005)
www.networkedplanet.com/download/tm-website-architecture.pdf

Techquilla
"TMShare – Topic Map Fragment Exchange in a Peer-to-Peer Application", Kal Ahmed
www.techquila.com/topicmapster.html

XTM TopicMaps.org
XML Topic Maps (XTM) 1.0
www.topicmaps.org/xtm/1.0/

### Sobre os autores

**Kal Ahmed** é cofundador da Networked Planet Limited (www.networkedplanet.com), desenvolvedor de ferramentas de mapas de tópicos e aplicativos baseados em mapas de tópicos para a plataforma .NET. Ele trabalha no gerenciamento de informações em SGML e XML há 10 anos, no desenvolvimento e na consultoria de softwares, e no kit de ferramentas do mapa de tópicos de Java de código-fonte aberto, TM4J, tendo contribuído para o desenvolvimento do padrão ISO.

**Graham Moore** é cofundador da Networked Planet Limited e trabalha há oito anos nas áreas de gerenciamento de informações, conteúdo e conhecimento como desenvolvedor, pesquisador e consultor. Foi CTO da STEP, vice-presidente de pesquisa e desenvolvimento na Empolis GmbH e cientista-chefe na Ontopia AS.


## Walt Disney World Swan Hotel
## May 14-18, 2006

# New Date — Same Incredible Content

**VSLive! Orlando arrives early this year** as weather concerns have moved the event to May. Microsoft insiders and industry veterans will return to Orlando for dynamic sessions, keynotes and product demonstrations on the latest innovations in development.

## VSLive! Brings You:

- Informative Keynotes from major industry players
- In-Depth Workshops on essential development topics
- A full slate of Microsoft-led sessions on .NET Day
- ASP Live! - Intensive server-side and Web development techniques for both VB and C# practitioners
- Smart Client Live! - Practical advice on smart-client development in VB and C#
- SQL Live! - Valuable tips on optimizing the performance and reliability of SQL Server
- An insider's tour of Windows Communication Foundation (aka Indigo)
- Virtual Tracks on a range of essential programming topics; and more

### Save the Dates - More Upcoming Events

**VSLive! Las Vegas**
- April

**VSLive! Toronto**
- April

**VSLive! New York**
- September

**VSLive! Boston**
- September

**VSLive! Chicago**
- September

FTP FAWCETTE TECHNICAL PUBLICATIONS

**Register by March 8th and SAVE $300**

## Call 800-848-5523 today or visit us online at www.vslive.com/orlando

# Projete e implemente uma fábrica de softwares

por Mauro Regio e Jack Greenfield


## Resumo

Nessa abordagem da interoperabilidade, compartilhamos a experiência obtida no projeto e na implementação de uma fábrica de softwares para sistemas de assistência médica com base no padrão HL 7 (Health Level Seven). Discutimos a visão de longo prazo e a prova de conceito de escopo reduzido desenvolvidos até agora. Também destacamos os desafios encontrados no nosso projeto e as oportunidades para ampliar o escopo da abordagem para setores diferentes e, em geral, as oportunidades para apoiar a colaboração de empresa para empresa.


O objetivo aqui é compartilhar a experiência obtida na hora de projetar e implementar uma fábrica de software baseada em HL7, um padrão de interoperabilidade entre as organizações de assistência médica. Começamos o trabalho há quase um ano com a especificação de alto nível da fábrica, produzindo uma primeira versão de seu esquema e da arquitetura de solução (consulte Recursos).

Em sua fase inicial, a fábrica se voltou para o projeto das portas de colaboração HL7, que são sistemas projetados para implantação à margem dos sistemas de TI das organizações de assistência médica, e permitir que os aplicativos de assistência médica colaborem em conformidade com os protocolos comerciais e técnicos padronizados na HL7 Versão 3, usando uma infra-estrutura de comunicação baseada no serviço da Web.

Na segunda fase, implementamos uma primeira versão – de escopo reduzido – da fábrica HL7 especificada na primeira fase. O foco dessa versão é o subconjunto dos recursos da porta de colaboração HL7 necessários para permitir a comunicação entre os aplicativos de assistência médica por meio de adaptadores do serviço da Web, em conformidade com os perfis do serviço da Web HL7 (consulte Recursos).

O escopo completo da fábrica, conforme especificado, também incluía o desenvolvimento dos adaptadores de integração dos aplicativos empresariais para conectar os aplicativos existentes às portas de colaboração e a orquestração das trocas de mensagens comerciais, percebendo uma colaboração em particular em prol dos aplicativos

**Figura 1** O contexto de produção de uma fábrica HL7

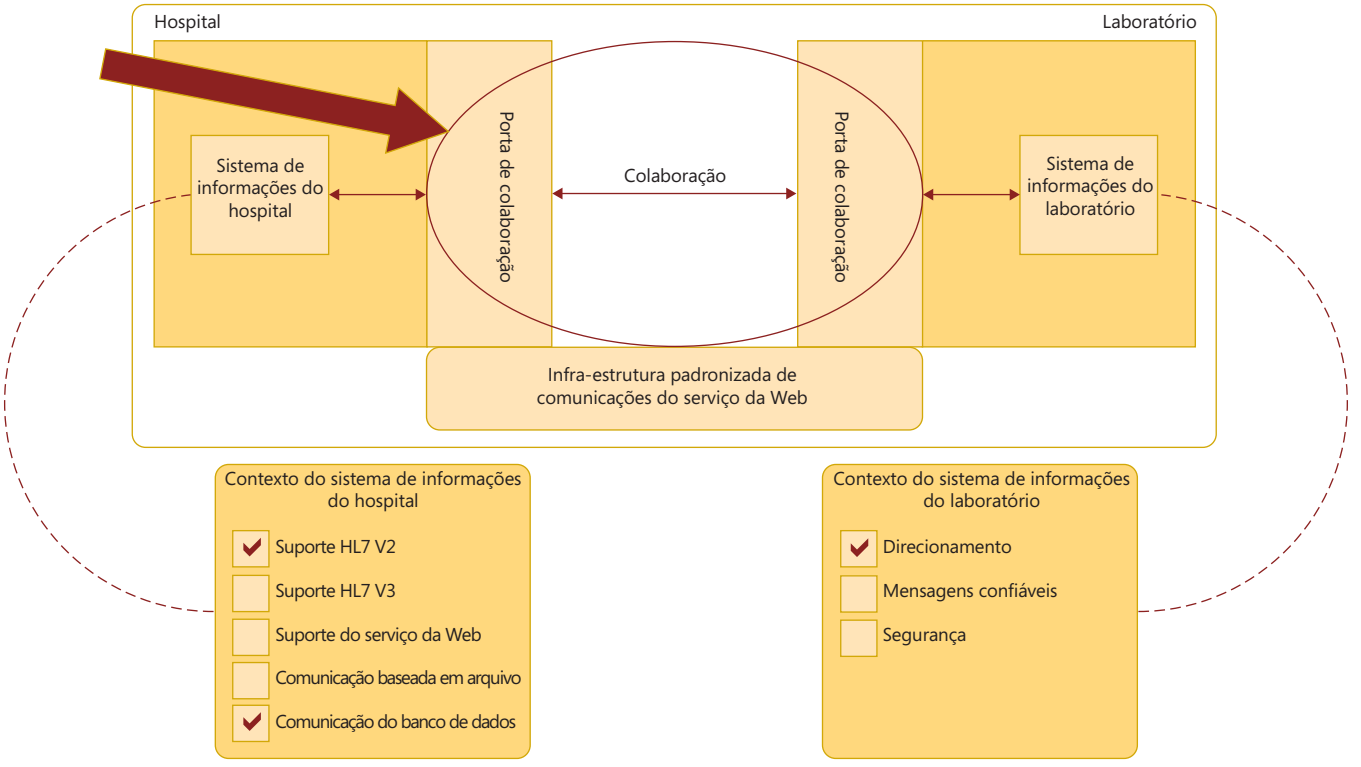
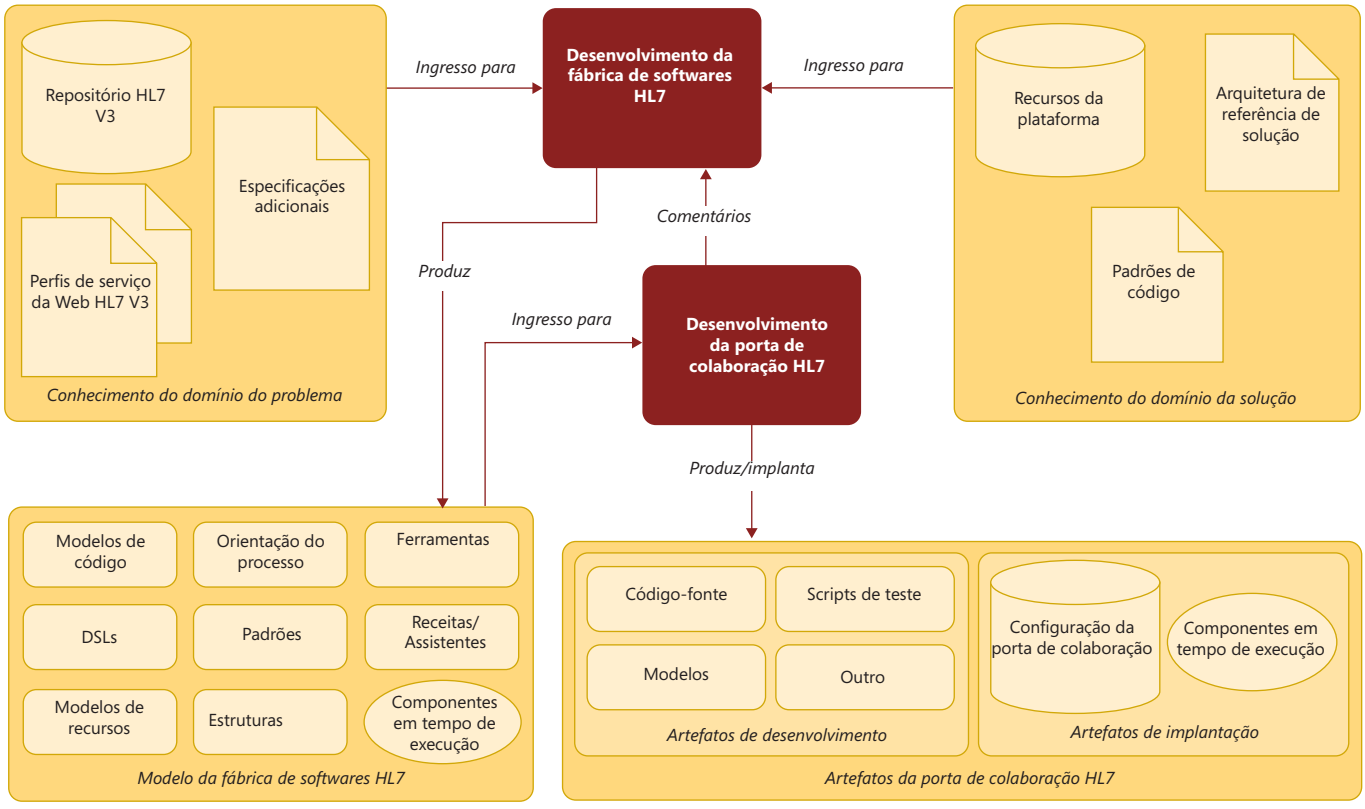


**Figura 2** O contexto de desenvolvimento de uma fábrica HL7




de linha de negócios que não foram projetados para a colaboração. Nossa experiência em projetar e desenvolver a fábrica HL7 mostra-se valiosa de duas perspectivas diferentes. Ao desenvolver a fábrica HL7, encontramos alguns desafios para desenvolver o esquema da fábrica, gerenciar a sua configuração, compreender como seriam usadas as linguagens específicas de domínio e promover as ferramentas disponíveis no momento no ambiente de desenvolvimento.

Ao mesmo tempo, percebemos que o escopo da fábrica poderia ser ampliado da colaboração entre os aplicativos de assistência médica baseados no HL7 para uma noção mais genérica de colaboração entre os aplicativos baseados em especificações padronizadas (ou compartilhadas).

Portanto, atualmente estamos em vias de generalizar a abordagem comprovada na implementação inicial da fábrica HL7 para projetar e desenvolver o que chamamos de fábrica de colaboração comercial.

## Fábricas de software

As fábricas de softwares usam conhecimento de domínio específico, arquiteturas de solução, ferramentas e outros ativos reutilizáveis para ajudar os usuários a produzir tipos específicos de soluções de softwares. As fábricas se baseiam em três idéias principais: um esquema de fábrica de software, um modelo de fábrica e um ambiente extensível de desenvolvimento.

A fábrica configura um ambiente extensível de desenvolvimento, como Eclipse, Borland JBuilder ou Microsoft VSTS (Visual Studio Team System), usando um pacote instalável chamado modelo de fábrica de software ou pacote de orientação. Quando configurado dessa maneira, o ambiente de desenvolvimento torna-se um mecanismo especializado que acelera o desenvolvimento de um tipo específico de solução de software, como uma interface de usuário ou uma camada de acesso de banco de dados, ou talvez um aplicativo inteiro em um domínio de negócio como a assistência médica ou a segurança privada.

O modelo de fábrica de software é organizado por um modelo chamado esquema de fábrica de software. O esquema define um ou mais pontos de vistas relevantes para os interessados na produção das soluções almejadas de softwares. Cada ponto de vista define os artefatos do ciclo de vida produzidos ou consumidos por seus interessados, as atividades que eles realizam em relação a esses artefatos e os ativos reutilizáveis para suportá-los na hora de realizar essas atividades.

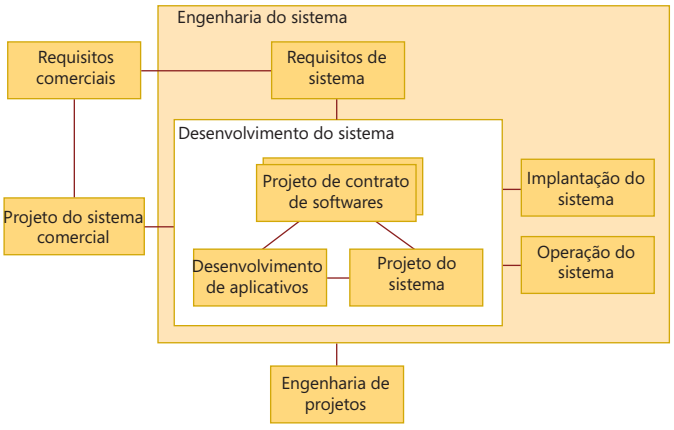
A metodologia da fábrica de softwares integra o MDD (Desenvolvimento Dirigido pelo Modelo), o CBD (Desenvolvimento Baseado em Componente) e as práticas ágeis de desenvolvimento, incluindo a utilização de padrões e linguagens de padrão com modelos, estruturas e ferramentas (consulte Recursos).

Para promover os modelos com eficiência para várias formas de automatização, as fábricas fazem um uso intenso das DSLs (Linguagens para Domínio Específico). A tecnologia DSL é muito mais recente do que a maioria das outras tecnologias usadas nas fábricas de softwares, baseando-se nas famílias de linguagens extensíveis. As ferramentas e estruturas de desenvolvimento de DSL já estão em desenvolvimento há algum tempo nos círculos acadêmicos, mas sua aparição na forma de negócio é muito recente (consulte Recursos).

A fábrica HL7 automatiza o desenvolvimento dos sistemas chamados de portas de colaboração, que permite a interoperação entre os sistemas no domínio de assistência médica. Especificamente, as soluções produzidas pela fábrica visam a:

- Perceber as interações definidas pelo padrão HL7 como trocas de informações que ocorrem entre os aplicativos em resposta aos eventos de acionador. Coletivamente, essas trocas suportam os objetivos de negócio de um caso de uso específico, como a realização da observação em laboratório. A fábrica automatiza a produção do código que implementa essas interações, minando as informações contidas no HL7 RIM (HL7 Reference Information Model).

**Figura 3** Ponto de vista da produção do sistema




- Permitir a colaboração de negócio de aplicativo para aplicativo expressa em termos dessas interações sobre uma infra-estrutura de serviço da Web, baseada em padrões abertos, que está em conformidade com um subconjunto dos perfis de serviço da Web do HL7 V3, a saber, os tópicos Básico, Direcionamento, Segurança e Mensagens confiáveis (consulte Recursos).
- Permitir a integração dos aplicativos novos ou existentes que não foram projetados: 1) para o HL7, versão 3; 2) para atender aos propósitos da colaboração comercial; e 3) para se comunicar em uma infra-estrutura de serviço da Web.

É importante entender a fábrica HL7 de duas perspectivas diferentes: de produção e desenvolvimento. No contexto de produção, os produtos finais da fábrica são as portas de colaboração HL7. Essas portas permitem automaticamente que aplicativos diferentes de assistência médica colaborem por trás do firewall usando os serviços da Web, já que pelo menos um dos aplicativos que participam da colaboração de negócio implantará uma porta de colaboração HL7; os outros aplicativos podem participar por outros meios, e todos os aplicativos que participam da colaboração de negócio estarão em conformidade com os padrões HL7 V3 para a troca de mensagens, ou de forma nativa, ou com a ajuda de uma porta de colaboração HL7.

## Fábrica por modelo

Para uma colaboração de negócio entre um hospital e um laboratório, a Figura 1 mostra onde as portas de colaboração HL7 se situam em relação aos sistemas que hospedam os aplicativos de interoperação. Observe que as portas de colaboração devem ser altamente configuráveis para permitir a expedição geral, permitindo também uma orquestração complexa no fluxo das mensagens. Assim, portas como as mostradas na Figura 1 são configuradas em termos dos detalhes técnicos para uma implementação e implantação específicas e em termos das definições de domínio e dos níveis de conformidade do padrão HL7.

No contexto do desenvolvimento, o objetivo da fábrica de softwares é acelerar a especificação e a implementação das portas de colaboração. A fábrica combina o conhecimento de domínio do problema fornecido pelo HL7 RIM e pelos perfis de serviços da Web, com o conhecimento da tecnologia de plataforma, da arquitetura de solução e do processo de desenvolvimento fornecido pela documentação da plataforma e pelos desenvolvedores da fábrica (consulte a Figura 2).

Como sugerido pela ilustração, esse conhecimento é empacotado em ativos numerosos, que formam, coletivamente, o modelo da fábrica. Colocado a questão de forma simples, o modelo de fábrica fornece tudo o que for necessário para construir uma porta de colaboração HL7, incluindo dados e artefatos de referência, tais como esquemas de mensagens; ferramentas, como geradores de adaptadores; e orientação do processo. O modelo de fábrica deve ser instalado em um IDE, a saber, Microsoft Visual Studio 2005 Team System, antes que possa ser usado para produzir e implantar portas de colaboração HL7.

Como foi anteriormente observado, o objetivo aqui é descrever o que aprendemos na especificação, no projeto e na implementação da fábrica HL7. Agrupamos as informações em duas categorias. A primeira classifica as lições aprendidas no desenvolvimento e na utilização da fábrica em geral; a segunda contém idéias obtidas em relação ao domínio principal e a como a fábrica pode ser generalizada para dar conta de uma gama mais ampla de domínios principais.

> "PRECISAMOS FORNECER AOS ARQUITETOS EMPRESARIAIS OS MODELOS E AS FERRAMENTAS NECESSÁRIOS PARA SUPORTAR O PROCESSO DE ESPECIFICAÇÃO"

O desenvolvimento e o gerenciamento do esquema da fábrica foram os desafios mais significativos desde a concepção do projeto até sua conclusão. Produzir uma versão inicial do esquema foi algo relativamente fácil (consulte Recursos). Nessa fase, é possível usar com eficácia uma abordagem baseada em grade, organizando os pontos de vistas relevantes em uma matriz bidimensional com o nível de abstração no eixo vertical e a fase do ciclo de vida no eixo horizontal.

No entanto, rapidamente nos demos conta de que a matriz bidimensional era uma representação bastante inadequada do esquema, especialmente para a versão de escopo completo da fábrica porque: a) o esquema era naturalmente multidimensional; b) a representação de matriz não captura as relações entre pontos de vistas não adjacentes; c) os gráficos dos pontos de vistas foram de tipos e profundidades diferentes e se desdobraram em gráficos aninhados de tipos e profundidades diferentes.

Não obstante, trabalhar com uma representação amorfa baseada em gráfico do esquema exigiu ferramentas que não estavam disponíveis. Portanto, implementamos o esquema de fábrica como um conjunto de projeções bidimensionais dos pontos de vistas relevantes. Cada uma dessas projeções – um gráfico em seu próprio sentido – detalha um aspecto específico da fábrica, projetando-o com eficácia a partir do esquema multidimensional da mesma maneira que um conjunto de tuplas é projetado de um armazenamento de dados multidimensionais para formar uma imagem bidimensional.

Por exemplo, as Figuras 3 e 4 mostram dois exemplos de pontos de vista, a saber, o ponto de vista do Desenvolvimento do sistema, e um aspecto particular dele: o Projeto de contrato de Software – a um nível inferior de abstração.

**Figura 4** Ponto de vista do projeto de contrato de softwares

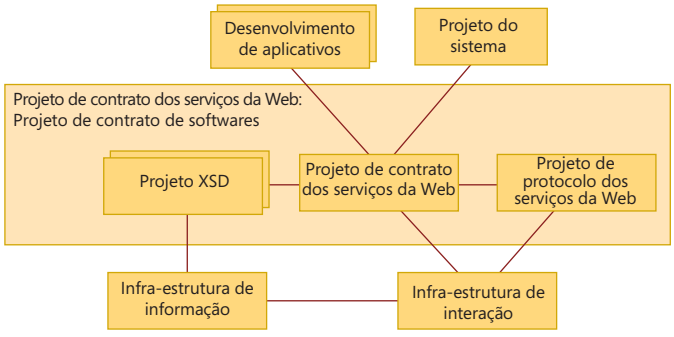
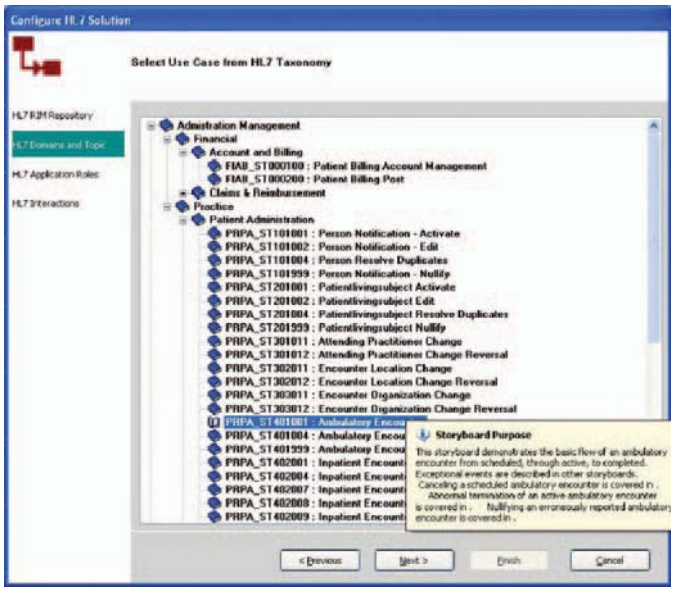


**Figura 5** A seleção de casos de uso do assistente de configuração




## Atribuindo tarefas

Essa abordagem nos permitiu produzir uma versão do esquema de fábrica que consideramos completa, ou seja, especificou de forma abrangente todos os artefatos e ferramentas necessários no modelo de fábrica para produzir os produtos da fábrica. No entanto, ela deixou a verificação do esquema para a inspeção humana, não nos permitindo usar o esquema como metadados para orientar a experiência do usuário dentro do IDE.

O gerenciamento de configuração foi outro aspecto desafiador do desenvolvimento de fábrica, de duas perspectivas diferentes.

Em primeiro lugar, tivemos que criar um esquema XML de configuração, que estaria completo em termos de permitir a expressão de todas as combinações válidas dos recursos aceitos e/ou das estratégias de implementação. O esquema XML foi idealizado à mão e rapidamente se tornou um fardo de manutenção, uma vez que foi altamente sensível às mudanças no esquema e no modelo de fábrica. Em segundo lugar, não tínhamos ferramentas no ambiente-alvo de desenvolvimento para suportar a configuração de uma instância específica da fábrica, ou a validação de tal configuração, durante o desenvolvimento do produto.

A especificação do processo de desenvolvimento do produto também foi um desafio na hora de desenvolver a fábrica. Não contávamos com nenhuma forma satisfatória de expressar formalmente o processo no modelo de fábrica, e, mais importante, nenhuma maneira de injetar o processo enquanto orientação prescritiva no ambiente de desenvolvimento.

Embora o ambiente-alvo de desenvolvimento não suporte a criação de tarefas nem a atribuição das tarefas aos membros da equipe de desenvolvimento, tínhamos que confiar em documentos em linguagem natural para descrever o processo de desenvolvimento, pois ainda não tínhamos determinado como aplicar os recursos de gerenciamento de tarefas a um processo de desenvolvimento de produtos baseado em fábrica. Em particular, ainda não tínhamos determinado como associar as tarefas com ativos específicos fornecidos pelo modelo de fábrica, como carregar essas tarefas do modelo ou como configurar as tarefas para um produto específico.

Também achamos bastante difícil decidir se devíamos fornecer ou não uma DSL bem-desenvolvida para a fase de levantamento de requisitos do desenvolvimento de produtos, em especial porque a Microsoft publicou um conjunto de ferramentas DSL sob o guarda-chuva de sua Software Factories Initiative.

Sabíamos que uma DSL completa não era estritamente necessária, pois as especificações relevantes do caso de uso já nos eram disponíveis no repositório HL7. O que realmente precisávamos era de um assistente sofisticado que ajudaria o usuário a escolher os casos de uso específicos (consulte a Figura 5), as funções do aplicativo, os padrões de interação de serviço (consulte a Figura 6) e outros elementos padronizados de dados, e a navegar pelo repositório.

Além disso, a tecnologia do projetista da DSL oriunda da Microsoft ainda estava muito crua quando começamos o projeto, e sua adoção teria acrescentado um risco significativo ao projeto. Embora soubéssemos que estávamos perdendo a oportunidade de fornecer mais automatização sofisticada, e que a alternativa – uma interface de usuário baseada em assistente – não seria relevante fora do escopo da fábrica, decidimos não usar a tecnologia DSL nessa fase do projeto.

Como agora temos uma previsão tecnológica muito mais consolidada e robusta das ferramentas DSL, podemos começar a experimentar as DSLs nas versões subseqüentes da fábrica. Além disso, mesmo que tivéssemos decidido criar outra interface de usuário baseada em assistente, provavelmente a teríamos projetado e implementado usando as ferramentas DSL, em vez de tê-la desenvolvido a partir do zero.

O GAT (Guidance Automation Toolkit), outro estágio de ativação de tecnologia no guarda-chuva da Software Factories Initiative da Microsoft, se mostrou bastante útil.

Em termos simples, o GAT é uma extensão do ambiente de desenvolvimento que facilita a criação de experiências valiosas e integradas do usuário em torno de ativos reutilizáveis como estruturas, componentes e padrões. Os pacotes resultantes de orientação são compostos de modelos, assistentes e receitas que ajudam os usuários a desenvolver soluções ao se manterem com a orientação arquitetural predefinida.

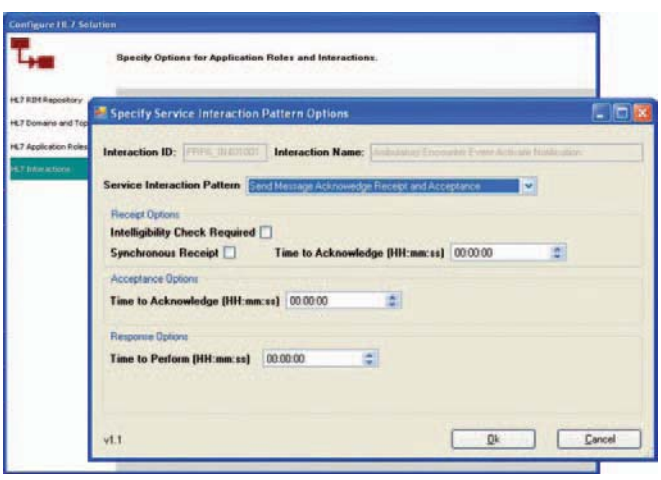
## Ampliar o escopo

No nosso projeto, usamos o GAT como meio de empacotar e fornecer o modelo de fábrica. Ele forneceu um modelo subjacente para o modelo que era muito mais valioso do que o que o próprio ambiente de desenvolvimento tinha para oferecer.

As receitas foram fornecidas para as atividades como a criação do Adaptador de serviços da Web HL7, a execução do assistente de configuração, a criação de contratos de serviços da Web e a automatização do processo de criação de código.

Infelizmente, a curva de aprendizagem do GAT é bastante acentuada. Suas opções de flexibilidade e personalização são limitadas, e sua integração com o IDE poderia ter sido melhor. No entanto, acreditamos que a adoção do GAT era uma decisão importante que nos

**Figura 6** A especificação dos padrões de interação de serviço do assistente de configuração



ajudou a garantir o sucesso do projeto e reduziu significativamente o tempo de desenvolvimento da fábrica.

Embora tivéssemos desenvolvido a fábrica para permitir a colaboração de uma empresa com a outra entre os aplicativos de assistência médica, rapidamente ficou claro que poderíamos aplicar uma fábrica assim a cenários semelhantes em outros setores. Conseguíamos entender que o escopo real da fábrica devia ser a colaboração de negócio em geral, usando especificações padronizadas ou predefinidas de interações, funções de aplicativos, eventos, perfis de serviços da Web e outros elementos de domínio, e não apenas a colaboração de negócio no contexto do HL7.

Em retrospectiva, a razão pela qual inicialmente desenvolvemos uma fábrica para o HL7 era que o padrão HL7 fornecia um conjunto completo de elementos de domínio bem-definidos e de fácil acesso. Naturalmente, muitos outras organizações de padrões, como a RosettaNet e a UNCEFACT, também investiram muitos esforços para especificar elementos de domínio a serem usados pelas empresas que desejam colaborar usando protocolos padronizados. O que é realmente interessante, no que tange aos protocolos de colaboração e às informações que trocam, é que esses padrões são muito parecidos entre si.

Assim, concluímos que seria possível não apenas desenvolver sistemas de colaboração baseados em padrões para outros setores, como também desenvolver uma fábrica de portas de colaboração que pode ser personalizada usando-se especificações de colaboração de negócio para outros setores da economia. Naturalmente, serão necessários vários mecanismos de importação, adaptadores e conversões de modelos que permitam o funcionamento de conceitos diferentes usados para descrever a colaboração de uma empresa a outra por entidades diferentes de padrões. No entanto, achamos que uma especificação genérica poderia ser usada para criar uma ponta entre essas diferenças por meio da configuração em uma fábrica genérica de colaboração comercial.

### Recursos

"A Software Factory Approach to HL7 Version 3 Solutions", Mauro Regio, Jack Greenfield, and Bernie Thuman (June 2005)

Domain-Specific Modeling with MetaEdit+
www.metacase.com

Health Level Seven
www.hl7.org

"Advanced Web Service Profiles", Roberto Ruggeri et al.
www.hl7.org/v3ballot/html/welcome/environment/

Institute for Software Integrated Systems
"Model-Integrated Computing"
www.isis.vanderbilt.edu

MSDN
Visual Studio Team System Developer Center
Microsoft Enterprise Framework and Tools Group
"Domain-Specific Language (DSL) Tools"
http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/

Microsoft Patterns and Practices Team
"Guidance Automation Toolkit (GAT)"
http://lab.msdn.microsoft.com/teamsystem/workshop/gat/

*Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Jack Greenfield et al. (Wiley, 2004)*

"Web Services Enablement for Healthcare HL7 Applications – Web Services Basic Profile Reference Implementation", Mauro Regio (August 2005)

Ao mesmo tempo, reconhecemos que uma fábrica mais genérica também pode ter um apelo bastante interessante junto aos desenvolvedores corporativos que estão construindo sistemas de colaboração de negócio dentro do firewall, e aos que desejam contar com uma abordagem mais formal para especificar e implementar esses sistemas para garantir o melhor alinhamento entre as metas comerciais e um portfólio de serviços de TI resultantes. No entanto, nesse caso, precisaríamos fornecer aos arquitetos empresariais os modelos e as ferramentas necessários para suportar o processo de especificação.

Percebemos que nossa fábrica deve aceitar a construção de portas de colaboração para várias plataformas de tecnologia. Suspeitamos que isso poderia ser realizado com padrões de implementação fornecidos com o modelo de fábrica para encerrar o vínculo entre a especificação das portas de colaboração e os detalhes de implementação específicos da plataforma.

## Assumindo a dianteira

Nossos planos são explorar a oportunidade descrita aqui para generalizar a fábrica HL7 e formar uma fábrica de colaboração comercial. Acreditamos que boa parte do trabalho necessário para conseguir essa generalização se encontrará nestas áreas:

- Fornecer modelos e ferramentas para suportar a especificação das colaborações de negócio, focando-se no esquema de informações, nos protocolos de troca de mensagens e documentos de negócio e, possivelmente, nas transações de negócio.
- Definir os mapeamentos entre os padrões industriais relevantes e nosso modelo interno de colaboração de negócio, e possivelmente as ferramentas para importar os elementos de domínio que eles definem.
- Apresentar outro nível de configuração na implementação da porta de colaboração que permitirá que os usuários da fábrica tenham como objetivo uma variedade mais ampla de plataformas de tecnologia.

Nossa experiência no desenvolvimento de uma fábrica para as portas de colaboração HL7 demonstra que precisamos definir estruturas, ferramentas e processos melhores para especificar o esquema da fábrica, gerenciar a configuração da fábrica de uma forma flexível e extensível e compreender melhor como e quando se devem usar as linguagens específicas de domínio. Ao mesmo tempo, as implementações iniciais de mecanismos de extensão, como o GAT e a DSL, provaram seu valor, preenchendo lacunas significativas na infra-estrutura da fábrica de softwares e apontando para inovações futuras nessa área.

Pretendemos continuar usando e aprimorando essas ferramentas na próxima versão da fábrica HL7, bem como na versão mais genérica e de vários setores, chamada de fábrica de colaboração comercial.•

### Sobre os autores

**Mauro Regio** é arquiteto de soluções industriais na equipe de estratégia de arquitetura do parceiro e do desenvolvedor na Microsoft, voltado para projetos e fábricas de softwares de integração de negócio de larga escala.

**Jack Greenfield** é arquiteto de softwares e ferramentas empresariais na Microsoft.

# Inteligência de Negócio Orientada a Serviços

por Sean Gordon, Robert Grigg, Michael Horne e Simon Thurman


## Resumo

Essa discussão analisa as semelhanças e diferenças entre BI (Inteligência de Negócio) e SO (Orientação a Serviços), dois paradigmas arquiteturais que se desenvolveram de forma independente. Definimos aqui uma estrutura arquitetural que promove os pontos fortes de BI e SO, ao mesmo tempo em que definimos princípios orientadores para garantir que as doutrinas fundamentais de cada uma das arquiteturas integrantes não estejam comprometidas.


O substantivo sinergia significa a ação combinada de entidades ou condições discretas, uma vez que o efeito total é maior do que a soma dos seus efeitos individuais. A SoBI (Inteligência de Negócio e Orientação a Serviços) constitui a sinergia dos paradigmas de BI e SO e descreve os padrões e a arquitetura para realizar essa sinergia, fornecendo uma estrutura de implementação recomendada pelas melhores práticas; a capacidade de se integrar ao nível arquitetural mais apropriado; a modelagem de dados de um projeto de BI dentro da estratégia de SO de deixar os sistemas de fontes implantados; e uma implementação comum para as transformações de dados e a lógica de dados: de dados para dados, de dados para serviços, de serviços para dados e de serviço para serviço.

A BI e a SO constituem paradigmas amplos. Vamos começar definindo BI. O data warehousing é uma disciplina ampla que permite a coleta, a consolidação e o armazenamento de dados para suportar a BI. Os componentes de um data warehouse bem-sucedido podem ser resumidos como coleta, limpeza e consolidação de dados ETL (Extract Transform and Load), e armazenamento de dados. A BI é a entrega das informações para suportar as necessidades de tomada de decisão da empresa. Pode ser descrita como o processo de aprimorar os dados na forma de informações e depois na forma de conhecimento.

**Figura 1** Visões da BI e da SO

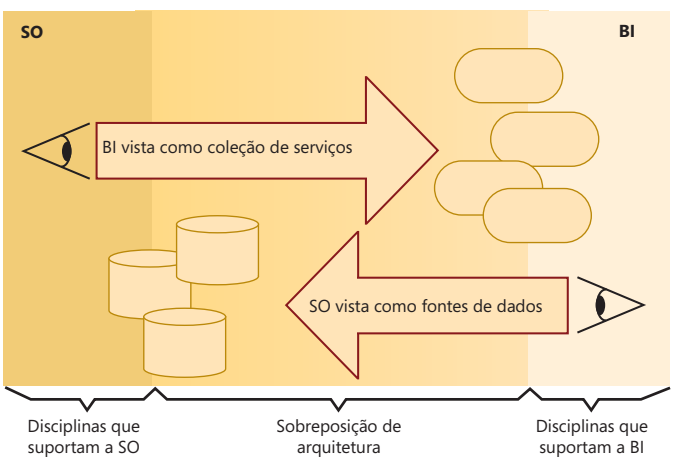



**Figura 2** Visão que a BI tem da SO

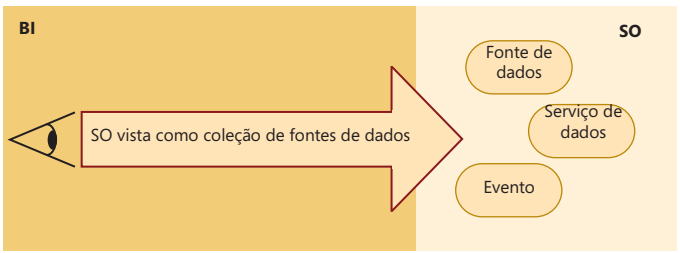
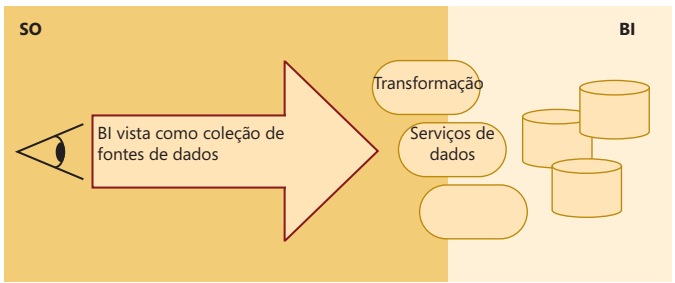



**Figura 3** Visão que a SO tem da BI




Cada sistema de BI tem uma meta específica, que se origina das exigências do negócio.

Para facilitar a leitura, o armazenamento de dados e a inteligência de negócio serão consolidados na sigla BI ao longo dessa discussão. A SO é um meio de construir aplicativos distribuídos; no seu nível mais abstrato, a SO enxerga tudo como prestador de serviços: dos aplicativos para as empresas, passando pelos dispositivos. Os prestadores de serviços expõem os recursos por meio das interfaces. Essas interfaces definem o contrato entre o chamador do serviço e o serviço em si. O consumidor de um serviço não se importa em saber como o serviço é implementado, apenas o que ele faz e como invocá-lo.

Os próprios serviços são os alicerces dos aplicativos orientados a serviço. Os serviços encapsulam os aspectos fundamentais da orientação ao serviço, ou seja, a separação entre a interface e a implementação. A SO é essencial para garantir a agilidade de negócio e a flexibilidade de TI prometidas pelos serviços da Web. Esses benefícios são entregues não apenas ao se visualizar a arquitetura de serviços da perspectiva tecnológica ou ao se adotar protocolos de serviços da Web, mas também ao se exigir a criação de um ambiente orientado ao serviço que se baseia em princípios específicos importantes.

A BI existe há anos; suas práticas são bem-estabelecidas, e as pessoas sentem-se confortáveis com os conceitos envolvidos no fornecimento de uma solução de BI. Para muitos, a BI é simplesmente a apresentação de informações de uma maneira oportuna por uma interface sofisticada de cliente. Para aqueles envolvidos no fornecimento de soluções de BI, esse aspecto da solução geral de BI é a ponta do iceberg. Abaixo do nível da água, há um enorme exercício no aprimoramento da qualidade dos dados e a integração dos dados entre aplicativos e sistemas corporativos distintos, e a consolidação desses dados em um armazém de dados. A integração dos dados tem predominantemente o maior custo em um projeto de BI.

## Convergência EAI e ETL

Até recentemente, a SO teve pouco ou nenhum papel para desenvolver no mundo da BI, principalmente porque a abordagem da SO na integração dos dados parecia cansativa e excessivamente complexa para uma comunidade acostumada a movimentar dados de qualquer volume conectando-se diretamente ao sistema de fontes no nível do banco de dados. Na BI, a integração dos dados é realizada por meio do processo ETL, que é a pedra angular de qualquer solução de BI, e as soluções de BI tendem a buscar a maneira mais direta e eficiente de realizá-la.

A EAI (Integração de Aplicativos Empresariais), como a BI, está presente há muitos anos. É um problema comum encontrado numa empresa em que os sistemas foram implementados ou desenvolvidos de uma maneira orgânica. A própria EAI pode ser definida como o compartilhamento do processo e dos dados entre aplicativos dentro da empresa. Especificamente, quando usamos o termo EAI, estamos nos referindo à integração dos sistemas dentro da empresa – por exemplo, integração de aplicativos, dados e processos.

A integração de aplicativo a aplicativo se refere à troca de dados e serviços entre aplicativos dentro da empresa. De forma especial, essa forma de integração ocorre com freqüência entre aplicativos que se encontram em plataformas tecnológicas diferentes, com base em arquiteturas diferentes. Em geral, a EAI é difícil e como de costume exige a conectividade entre plataformas tecnológicas heterogêneas; envolve regras e processos comerciais complexos, além de processos comerciais de longo prazo nos quais as unidades lógicas de trabalho podem se estender por dias ou semanas à medida que se movimentam por processos diferentes dentro da organização; e em geral é orientada pela necessidade de ampliar/aprimorar um negócio ou processo automatizado existente ou introduz um processo de negócio automatizado inteiramente novo.

As soluções para os problemas de EAI podem envolver a resolução do problema de integração dos aplicativos em uma série de níveis arquiteturais diferentes, como o de dados, aplicativos, processo e assim por diante. Dessa maneira, tanto a ETL quanto a SO podem fazer parte de uma solução EAI. De muitas maneiras, a SO se desenvolveu a partir da necessidade de encontrar soluções comuns, abertas e interoperáveis para o problema da EAI.

A ETL tradicional é um processo orientado por lotes que se concentra na integração dos dados durante o tempo de inatividade do negócio. No mercado conectado de hoje, a empresa não tem um tempo tranqüilo para que esse processo ocorra. O conjunto de dados corporativos tem o potencial de aumentar significativamente à medida que iniciativas como a seqüência de cliques, o comércio eletrônico e a RFID são cada vez mais utilizados, e a ETL deve ser flexível o suficiente para emergir de suas raízes orientadas a lotes e fornecer dados em uma base orientada a eventos.

Em uma solução SO, esses eventos são prontamente roteados, consumidos e integrados como parte de uma arquitetura orientada a eventos. A BI se desenvolveu da incapacidade dos sistemas operacionais de lidar com eficiência com as capacidades analíticas (agregação, tendência, exceção, etc.), daí que sistemas distintos de TI foram desenvolvidos para atender a essa necessidade. Essa divisão artificial não é mais aceitável; as empresas de hoje estão buscando cada vez mais utilizar as capacidades

**Figura 4** SO como fonte de dados para a BI

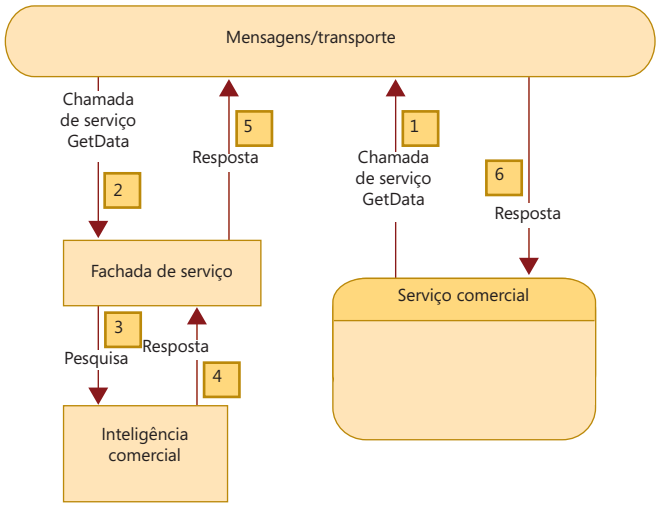
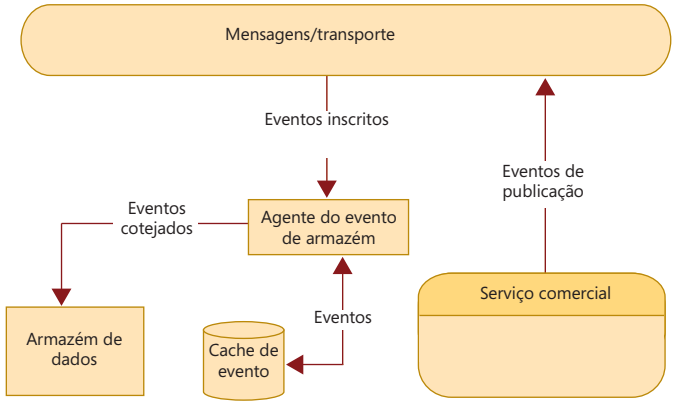



**Figura 5** BI consumindo os eventos de SO




analíticas para orientar suas decisões operacionais – por exemplo, na descoberta de um cartão de crédito suspeito fraudado no checkout, com base nos dados históricos extraídos em padrões fraudulentos.

As organizações também estão cada vez mais ansiosas para desbloquear esses dados e disponibilizá-los mais amplamente a outras partes da organização, para um público maior de usuários e ferramentas. Tradicionalmente, o acesso às informações da BI sempre exigiu acesso a um conjunto específico de ferramentas de manipulação de dados. No atual ambiente de SO, a meta deve ser abrir esses dados organizacionais para um público mais amplo e permitir que o valor dos dados seja percebido de forma mais abrangente.

## Uma variedade mais ampla

À medida que aumentam a variedade e o número de fontes de dados considerados dentro do escopo da BI, aumenta também a complexidade em potencial da solução ETL exigida. Por exemplo, os serviços da Web, o RSS e os dados não estruturados e semi-estruturados são fontes de dados que agora se enquadram sob o guarda-chuva da integração de dados, mas não estão tradicionalmente associados a ETL. Com a adoção disseminada dos princípios orientados ao serviço e as tecnologias associadas de ativação, o acesso a uma maior variedade de sistemas torna-se possível, desbloqueando uma variedade muito mais ampla de dados comerciais.

As organizações continuam a escrutinar a economia da integração de dados em cada projeto, afetando os produtos que selecionam, e isso está gerando uma mudança no cenário da integração de dados e uma mudança relacionada na funcionalidade fornecida por software de ETL ou EAI. Os fornecedores precisam aumentar a flexibilidade e a funcionalidade oferecidas por suas plataformas em resposta a esses novos desafios. O resultado líquido para o cliente é um conjunto de ferramentas com uma gama cada vez mais integrada de funcionalidades.

Os usuários familiarizados com o conjunto de produtos da Microsoft não podem ter deixado de perceber esse padrão. O BizTalk foi desenvolvido no espaço da EAI e da filosofia B2B (Business-to-business), mas pode ser aplicado em alguns cenários de ETL. Os DTSs (Serviços de Transformação de Dados), a ferramenta ETL da Microsoft que faz parte da plataforma SQL Server 2000, foram desenvolvidos a partir de um cenário de ETL. Não é nenhuma coincidência o fato de que a versão SQL Server 2005 desse projeto tenha sido rearquitetada para suportar uma variedade mais ampla de cenários de integração, tendo sido rebatizada de Integration Services para enfatizar isso.

Embora as SOAs e as arquiteturas de BI tenham evoluído separadamente e incluído tecnologias e disciplinas específicas aos seus próprios objetivos arquiteturais, muitas das tecnologias que elas utilizam se sobrepõem. Também há um claro mapeamento entre os conceitos utilizados, e cada um deles consegue ver o outro em seus próprios termos. Chamamos isso de "visões do outro lado", e a visão geral pode ser vista na Figura 1.

Vamos discutir cada cenário em mais detalhes.Da perspectiva da BI, é possível enxergar um aplicativo de SO como uma coleção de fontes de dados e de eventos. Há dois modos principais nos quais um serviço pode funcionar como fonte de dados em um contexto de BI: serviço como o provedor dos dados mediante solicitação e serviço como editor de eventos que são de interesse (consulte a Figura 2). Em ambos os cenários, os tamanhos da mensagem são pequenos. A solução para a transferência e a transformação dos dados em larga escala ainda será por meio das técnicas normais de importação do armazém de dados, tais como a ETL. Essas mensagens fisicamente grandes não são o domínio normal da SO.

Do ponto de vista da SO, a BI pode ser vista como uma coleção de serviços. Da perspectiva da SO, uma fonte de dados pode ser prontamente exibida como serviço com a inserção de uma camada simples de fachada que recebe a solicitação de serviço do barramento de serviço e chama a solicitação apropriada. A "fachada" então transforma os resultados da solicitação (se necessário) no esquema de dados e retorna os resultados ao chamador (consulte a Figura 3).

A arquitetura SoBI disponibiliza os dados de BI no armazém de dados como serviço para outros aplicativos dentro da arquitetura. Essa disponibilidade fornece aos aplicativos uma forma tranqüila de acessar os dados consolidados para suportar as exigências da BI. Dessa forma, a arquitetura de BI torna-se um componente integrado da arquitetura de aplicativos da SO. Observe que haverá ocasiões em que o tipo de dados que é necessário do sistema é puramente de natureza da inteligência de negócio , tais como a exportação de dados de larga escala. Nesses cenários, a abordagem da interface do serviço não será adequada.

**Figura 6** Tamanho da mensagem versus volume da mensagem

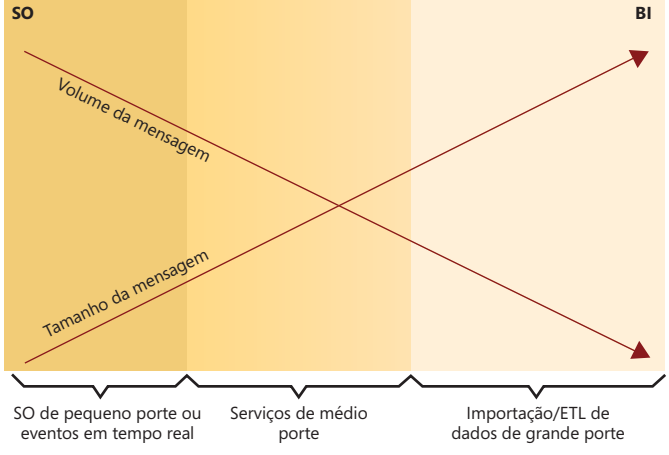



**Tabela 1** Benefícios da SO e da BI

| SO (Orientação ao Serviço) | BI (Inteligência Comercial) |
|---|---|
| Mais bem adequada à integração de aplicativo para aplicativo e bem ajustada aos eventos de baixo volume e baixa freqüência | Mais bem adequada à integração de dados para dados e capaz de lidar com grandes volumes de dados |
| Fornece uma plataforma operacional, define com firmeza os formatos e as estruturas de dados e encapsula e abstrai a funcionalidade | Fornece um modelo combinado dos dados empresariais e fornece as bases para as decisões comerciais, além da capacidade de fazer qualquer pergunta sobre os dados |
| Suporta a reutilização dos componentes empresariais e permite a mudança ágil nos processos comerciais | Ferramentas e mecanismos bons para transformar os dados |

## Provisão de dados orientada ao serviço

Da perspectiva da inteligência comercial, um serviço pode ser prontamente exibido como fonte de dados com a inclusão de uma camada simples de fachada que fornece um mapeamento entre a interface de BI e a interface exposta pelo serviço. A "fachada" então transforma os resultados da chamada do esquema de dados usado no barramento de serviço no formato de dados esperado pela plataforma de BI , devolvendo os resultados ao chamador (consulte a Figura 4).

Alguns serviços expõem as informações por meio de eventos que são publicados quando ocorre uma mudança interessante no serviço. Outros serviços e aplicativos na organização podem se inscrever nos eventos publicados pelos serviços. Integrar serviços de publicação de eventos em uma plataforma de BI é algo que se consegue com utilização de um agente que coteja os eventos inscritos e os transfere periodicamente em grandes quantidades para a plataforma de BI (consulte a Figura 5).

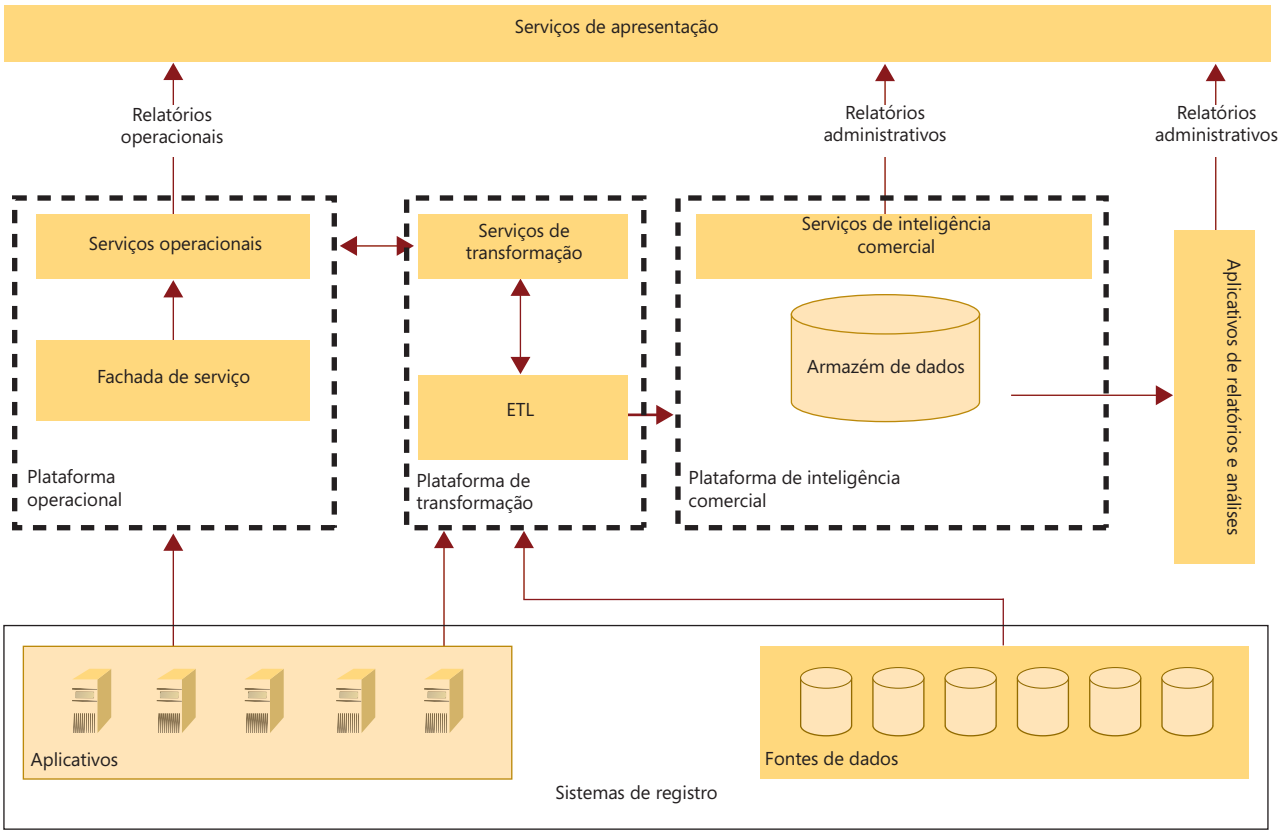
Um dos desafios no desenvolvimento da SoBI era encontrar uma abordagem que promovesse os principais pontos fortes de cada arquitetura e identificasse a área em que a integração causava mais desafios. Vamos dar uma olhada em alguns dos principais desafios inerentes à implementação da BI seguindo os princípios da SO. Esses desafios geralmente surgem devido às exigências específicas das quais cada arquitetura precisava dar conta no seu desenvolvimento.

A Figura 6 mostra as diferenças na granularidade dos dados que separa as abordagens da BI e da SO. Em pontos extremos, temos mensagens ou eventos de pequeno porte, que estão naturalmente no espaço de evento da SO. Essas mensagens de pequeno porte não são consumidas de forma imediata ou eficiente em uma arquitetura de BI sem a utilização de agentes de eventos para cotejar esses eventos e importá-los para o armazém de dados em uma base regular. Os dados de grande porte, ou seja, a importação em massa ou o movimento de grandes quantidades de dados, são manuseados com mais eficiência por meio das técnicas de armazém de dados, como a ETL. Os serviços em um sistema de SO que expõem grandes quantidades de dados são ineficientes de usar e raramente implementados. No meio do caminho, temos os serviços típicos de médio porte, definidos para consumir dados suficientes para atender a uma exigência em particular. Esse meio do caminho é a chave para o valor agregado da SoBI.

Os aplicativos orientados ao serviços apresentam um vínculo fraco entre seus serviços integrantes. Esse vínculo fraco é um dos princípios fundamentais da SO e suporta o desenvolvimento de aplicativos ágeis/flexíveis que podem se adaptar à mudança comercial. Consulte a Tabela 1 para ver os benefícios específicos da SO e da BI.

As soluções de BI estão firmemente vinculadas às fontes de dados que alimentam o armazém de dados e aos aplicativos que o utilizam. A BI evoluiu de um ambiente centrado em lotes no qual a ETL é

**Figura 7** A estrutura SoBI




usada como meio para consumir e consolidar diretamente grandes quantidades de dados do sistema de fonte, seguindo um cronograma conhecido de preenchimento do armazém de dados.

A SO exige que a interface da mensagem e os formatos das mensagens de entrada e da resposta eventual estejam firmemente definidos. Com efeito, ao se expor os serviços que descrevem as capacidades comerciais, as questões que podem ser feitas em um aplicativo de SO são conhecidas de antemão. No entanto, também há um aspecto desconhecido que se relaciona à exposição dos serviços que são agregados ou orquestrados por outro aplicativo. A BI está preocupada com a capacidade de permitir que os aplicativos façam qualquer pergunta ao armazém de dados dentro dos limites do modelo de armazém de dados. A pergunta, ou o tamanho, o conteúdo e o formato do resultado não são conhecidos até que a questão seja feita.

## A SoBI vence

Uma abordagem SO é mais adequada para a exposição dos serviços que encapsulam as capacidades ou serviços comerciais que publicam eventos de interesse para outros sistemas. A BI fornece um ambiente fechado para satisfazer as exigências de informações da empresa. A BI permite que o consumidor dos dados os visualize de muitas e diferentes maneiras potencialmente novas. Essa flexibilidade fornece a capacidade de identificar tendências e relações que podem ser negligenciadas.

Anteriormente apresentamos os principais pontos fortes de cada um dos paradigmas, e eles podem ser vistos como questões fundamentais caso sejam vistos puramente da posição do outro paradigma. Agora vamos revisitar esses desafios e ver quais benefícios a SoBI pode trazer.

*A funcionalidade da ETL na superfície do warehouse para a SoBI* permite que eventos comerciais interessantes, apreendidos durante o processo de ETL, sejam publicados como eventos. Vamos considerar alguns exemplos.

*Integridade referencial.* A SoBI pode fornecer agregações da posição "agora". Por exemplo, os dados que não se encontram no sistema de origem podem ser adicionados no armazém de dados como espaço reservado, a fim de manter a integridade dos dados no banco. Esse espaço reservado pode ocorrer quando sistemas distintos de fontes fornecem dados em momentos diferentes (por exemplo, valores transacionais fornecidas antes que as informações associadas de referência sejam recebidas). Esses dados serão armazenados como dados reservados privados dentro do serviço da BI. É possível enviar uma notificação para o sistema de fonte para garantir que não tenha ocorrido um erro, e quando os dados de referência tornam-se disponíveis o armazém de dados estará novamente em sintonia com o sistema de registro.

*Serviço de validação única.* A funcionalidade necessária para a validação durante o estágio da ETL pode ser exposta por meio da estrutura SoBI para utilização em outras partes da empresa.

*Atualizações no momento certo.* A principal vantagem aqui para a BI é que a adoção da ETL da perspectiva da SO pode permitir inserções reais em tempo suficiente no armazém de dados e, portanto, o armazenamento de dados potencialmente em tempo real. Além disso, o melhor suporte aos eventos e à integração orientada a eventos pode fornecer à BI um mecanismo muito melhor para invocar a ETL do que os métodos tradicionais, como o cronograma agendado ou a análise persistente de um diretório conhecido por um arquivo indicador.

*Esquema comum de negócio.* Dentro de qualquer organização, pode haver múltiplos sistemas que mantêm as informações sobre as mesmas entidades e que guardam essas informações em diferentes formatos. Há sinergias claras entre esses cenários:

- Desenvolver o modelo lógico de dados para o armazém de dados é essencial para qualquer iniciativa de BI. O modelo de dados constitui o produto final dos esforços para consolidar os dados

**Tabela 2** Princípios da SoBI

| | BI | SO |
|---|---|---|
| **É...** | ...a versão única da verdade para os dados de BI. | ...a abordagem arquitetural para a integração de aplicativos. |
| **Ela...** | ...fornecerá acesso aberto aos serviços de dados, suporte à análise ad hoc e aos relatórios *pré-elaborados* de administração, consolidação dos dados de sistemas de fontes distintas e suporte aos dados de referência. | ...fornecerá a integração de aplicativo para aplicativo e algumas inserções de eventos no DW; descreverá os serviços prestados e as mensagens transmitidas; atenderá às exigências operacionais e fornecerá os serviços de infra-estrutura para todos os aplicativos. |
| **Ela não...** | ...se tornará um terreno baldio para todos os dados, se tornará o proprietário dos dados, será a fonte de dados padrão para outros aplicativos e fornecerá a relatórios operacionais. | ...será usada em todas as circunstâncias e substituirá as interfaces de importação de dados |

dos sistemas de fontes distintos. A estrutura do modelo de dados orienta o exercício de transformação que ocorre como parte do preenchimento do armazém de dados e, conseqüentemente, permite que o armazém de dados forneça a versão única da verdade para as informações de gerenciamento.

- Para qualquer serviço de agregação de entidade dentro de um projeto de SO, é importante conseguir consenso sobre os significados comuns para as entidades nas quais os serviços vão operar. Esse consentimento é referido como consolidação do esquema, sendo o processo de criar esquemas de dados principais que contêm um superconjunto das informações para descrever as entidades no sistema com detalhes suficientes para que serviços diferentes possam localizar os dados de que necessitam.

Outro serviço de entidade que entra em sintonia com essa abordagem é a capacidade de expor os dados de referência. De acordo com Easwaran G. Nadhan, diretor-geral do EDS, "fica claro com as experiências em um número (relativamente pequeno) de organizações que passaram a adotar as SOAs agressivamente que coordenar os dados de referência é o primeiro passo necessário para se conseguir a orientação de serviço" (consulte Recursos).

## Uma verdade

*Uma versão da verdade.* Dando uma olhada na funcionalidade oferecida por duas arquiteturas holisticamente, a SoBI nos fornece a oportunidade de consolidar os dados operacionais e de BI sem a necessidade de transferir fisicamente todos os dados operacionais para a plataforma de BI, que é uma abordagem comum para fornecer a "versão única da verdade" em um projeto de BI. Adotando a escolha arquitetural mais apropriada, os dados operacionais podem ser deixados no lugar, mas ainda estar disponíveis para a estrutura da SoBI como serviço, caso surja a necessidade de acessá-los ou consultá-los.

Por exemplo, em um ambiente de TI interessado na análise de incidentes de segurança ou de saúde, a SoBI permitiria que os detalhes factuais de cada incidente fossem transferidos para o armazém de dados – ou seja, o fato de que um acidente aconteceu, onde aconteceu e a classificação desse incidente –, o que permitiria que toda a análise e a agregação apropriadas fossem realizadas como esperado pela plataforma de BI.

A vantagem da SoBI é que ela fornece um meio de ainda acessar os detalhes transacionais no sistema de registro (por exemplo, o texto em formato livre que acompanha o incidente, que descreve em detalhes as circunstâncias em volta do acontecimento). Esse acesso é geralmente conhecido como "detalhamento" na BI, e para realizá-lo todos os dados relevantes à exigência são tradicionalmente transferidos para o armazém de dados.

Na verdade, talvez não se permita (por razões de proteção de dados) ou não se prefira (aumenta o fardo da ETL e as exigências de armazenamento do armazém de dados para mantê-los, os quais, por definição, são operacionais por natureza) fazer isso, tendo o potencial de acrescentar pressão extra à plataforma de BI ao se tornar a verdadeira fonte para todos os dados de incidente, muito embora nunca esteja atualizada.

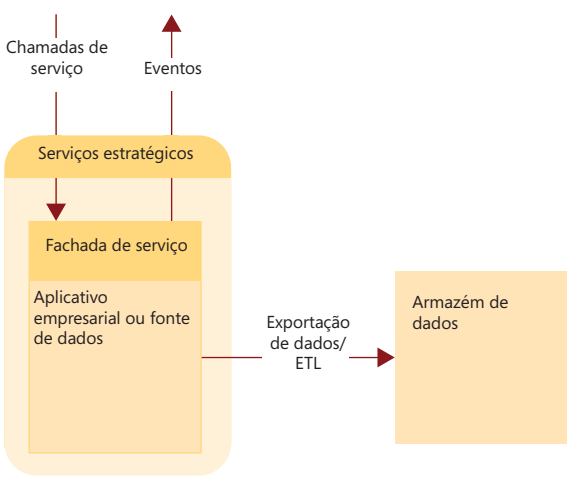
*Serviços comerciais.* A combinação da habilidade do BI trabalhar com tempo real suficiente e da capacidade de deixar os dados operacionais no lugar nos dá a oportunidade de desenvolver um serviço valioso em torno da estrutura SoBI que não seria possível caso estivéssemos trabalhando em apenas uma das arquiteturas integrantes. Pense em um sistema de detecção de fraude no varejo. Na BI, temos as ferramentas para desenvolver um mecanismo analítico capaz de buscar padrões em quantidades potencialmente enormes de dados transacionais, resultando em uma lista de cartões de créditos suspeitos. A adoção de princípios de SO nos permite oferecer um serviço que fornece os detalhes de cartões de créditos em utilização no nosso armazenamento à medida que são descartados. A arquitetura de SoBI permite que esse serviço seja consumido pelo componente de BI da plataforma e o analise em relação à lista conhecida de cartões suspeitos, e portanto responda imediatamente caso seja detectada uma transação potencialmente suspeita.

## Uma nova geração de serviços comerciais

*Agregação de dados transacionais* e históricos como serviço. Dado um serviço exposto por meio da estrutura de SoBI que agora pode agregar com perfeição os dados transacionais atuais e os dados históricos do armazém, é possível fornecer suporte a uma nova geração de serviços comerciais. Um exemplo seria *mudar lentamente as dimensões,* que é onde um valor como o nome do cliente muda com o tempo. Obviamente, essas informações estão disponíveis no armazém de dados, por isso, se houver a exigência de expor um serviço de entidade que dá uma visão única do cliente, pode-se conseguir isso com mais precisão e facilidade.

*Trazer os padrões de abstração de interface para a BI.* A capacidade de usar o padrão de abstração de interface na funcionalidade da BI torna a funcionalidade e os dados mais acessíveis aos aplicativos de linha de negócios, permitindo-se a exposição de regras comerciais complexas geralmente escondidas na camada da ETL.

**Figura 8** Dimensionamento e flexibilidade ideais



*Limpeza e consolidação.* Os dados serão alterados por motivos de consistência e integridade. Quando essa mudança envolve uma operação de mapeamento, este será disponibilizado para a arquitetura como serviço. Quando essa mudança envolve a correção dos dados, os detalhes dessa correção serão repassados ao sistema de registro por meio de uma solicitação de mudança ao serviço de posse, ou seja, o processo de ETL não pode alterar os dados uma vez que não é o proprietário. Por sua vez, esse processo obviamente promove o aperfeiçoamento da qualidade dos dados.

*Obter uma visão consistente e* de múltiplos sistemas do produto. Durante o estágio de ETL, os dados do mesmo produto (ou entidade) podem exigir transformação, para que possam ser armazenados de uma forma consistente. Com o serviço ativando o acesso aos dados, a organização pode expor uma única visão comum de um produto.

*Mapeamentos disponíveis como serviços.* Como se observou anteriormente, o mapeamento é uma exigência fundamental no estágio de ETL. A estrutura SoBI permite que a funcionalidade do mapeamento seja exposta como serviço para outros usos dentro da organização. Esses usos incluem cenários de referências empresariais e de EAI. Essa disponibilidade desse serviço também pode ser usada para promover transformações otimizadas.

*Cálculo.* O armazém de dados geralmente é usado para armazenar valores pré-calculados e suportar as exigências da inteligência comercial. Por exemplo, os dados de vendas e de previsões podem ser mantidos em sistemas físicos diferentes. A consolidação dos dados desses sistemas no armazém de dados nos permite calcular e armazenar os índices reais e os da previsão, permitindo análises e relatórios de alto desempenho. A lógica de negócio usada para definir esses cálculos geralmente interessa a outras partes da empresa, por isso o cálculo que fornece respaldo a essa invenção dos dados no armazém será disponibilizado para a arquitetura SoBI como serviço.

*Fornecer uma estratégia para a integração.* Acredita-se que um dos resultados da estrutura SoBI é a capacidade, no nível arquitetural, de fornecer uma estrutura para os futuros cenários de integração.

*Conformidade/auditoria.* A aplicação da estrutura SoBI exige a adesão a um processo formal de governança. Entre os exemplos estão a identificação do sistema de registro ou do proprietário dos dados operacionais, e a definição das mensagens que descrevem as exigências funcionais e dos dados. Dado que apenas o proprietário dos dados pode fazer uma mudança neles, outros sistemas simplesmente fazem uma solicitação de mudança, e a auditoria pode ser executada num ponto único.

*Agregação.* Para suportar tempos rápidos de resposta, os dados no armazém são geralmente pré-agregados. Por exemplo, o armazém de dados pode conter dados relacionados às vendas em um nível individual de transação, mas a maioria dos relatórios da administração pode exigir que os totais sejam apresentados mensalmente. Nesse caso, é mais econômico listar as (potencialmente milhares de) transações individuais em um nível mais apropriado para as pesquisas conhecidas e armazenar os resultados em um conjunto, evitando a necessidade de se fazer pesquisas conhecidas para realizar a ação de agregação no momento da pesquisa. Quando essas agregações forem criadas, elas serão disponibilizadas para a arquitetura como um serviço.

## Qualidade dos dados da empresa

A maioria das organizações sofre com o problema da compatibilidade entre seus sistemas de TI. Esse problema é ainda mais visível quando ocorrem fusões. Não existe razão pela qual os sistemas de empresas completamente diferentes sejam consistentes ou alinhados, ou satisfaçam um projeto unificado. Uma iniciativa de BI precisa resolver os problemas de sistemas diferentes, dos silos de dados e da incompatibilidade dos dados por meio de iniciativas de qualidade nos dados. Caso os usuários da empresa não confiem nas informações que lhe são apresentadas, eles não usarão o sistema.

**Figura 9** Ordenando eventos para a integração

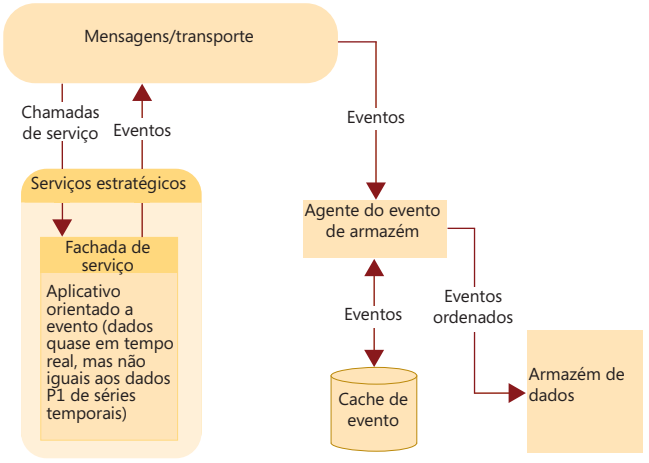
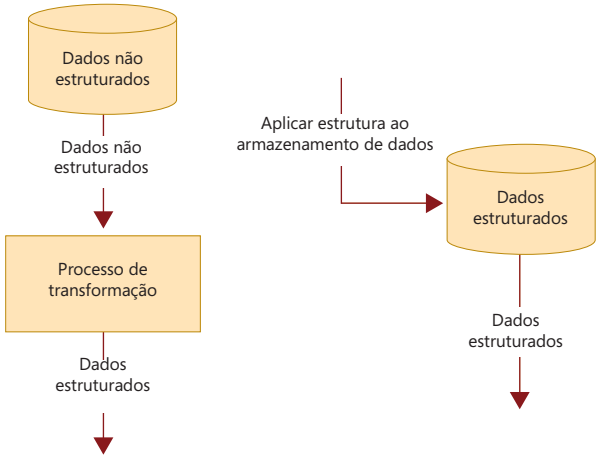



**Figura 10** Atualizando as fontes de dados não empresariais

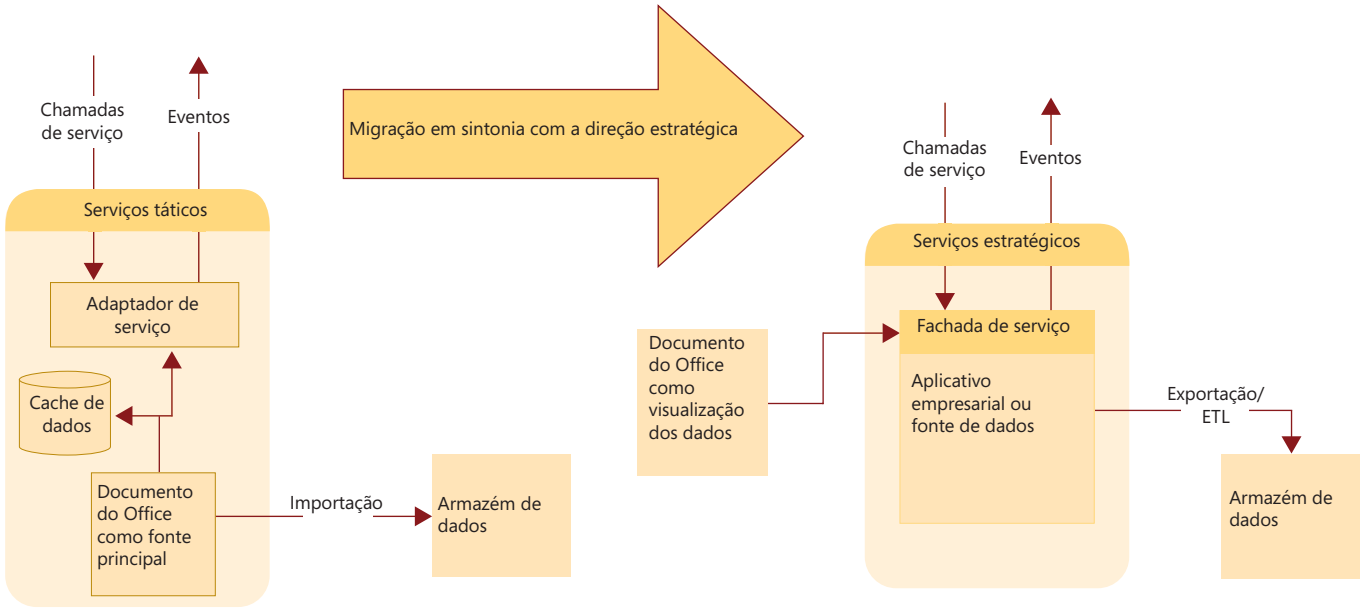



Melhorar a qualidade dos dados não é simplesmente um processo de resolver problemas com elementos individuais de dados; trata-se de projetar um processo de negócio que melhore o gerenciamento e a qualidade dos dados como recurso empresarial. Em um aplicativo de SO, os dados fornecidos por um serviço são controlados por meio de encapsulamento por ele realizado, sendo publicados apenas em um formato específico que corresponda ao esquema de dados definido pelo serviço.

Consegue-se o acesso aos dados por meio das mensagens que o serviço publica. O serviço é responsável apenas pela manutenção da integridade dos dados, uma vez que constitui o único mecanismo que manipula diretamente os dados. O esquema que define as entidades nas mensagens de serviço e a definição das próprias mensagens melhoram muito o aspecto da qualidade dos dados de qualquer solução, devido à capacidade de testar automaticamente as mensagens para ver se há conformidade com o esquema ou o contrato de mensagens suportados por um serviço.

Vamos dar uma olhada no exemplo de estrutura SoBI. É importante que a estrutura SoBI defina e diferencie claramente os diferentes tipos de dados e os proprietários associados desses dados, o que garante a integridade pois os dados podem ser reparados em apenas um lugar. Também permite que os dados sejam dispostos de acordo com o uso apropriado. A Figura 7 mostra a arquitetura de dados de alto nível para a estrutura SoBI. O objetivo dessa arquitetura é destacar como tipos diferentes de dados serão tratados na solução.

**Figura 11** Visualizações das informações




Como você pode ver, os sistemas de registro são integrados no armazém de dados por meio de métodos ETL tradicionais, com a exceção de que os serviços de transformação normalmente contidos no processo ETL são exibidos para reutilização por parte dos serviços tradicionais. Essa reutilização permite que as informações sejam integradas no armazém de dados e expostas por meio de fachadas de serviços em um esquema comum à medida que os serviços de transformação são compartilhados por ambos os mecanismos.

Também se reconhece que nem todas as informações no armazém de dados podem ser expostas por meio da interface de serviço e que ainda pode haver uma pequena população de usuários na organização que vão precisar de acesso direto ao armazém de dados para realizar uma análise complexa e ad hoc. Vamos dar uma olhada em mais detalhes nos fatores que influenciam a implementação bem-sucedida da estrutura SoBI.

Num alto nível, a orientação arquitetural para a aplicação das peças integrantes da SoBI pode ser resumida como mostrado na Tabela 2. Entre os fatores de sucesso para o projeto SoBI estão:

- *Governança.* É pouco provável que um projeto SoBI seja bem-sucedido sem um processo associado de gerenciamento de mudança organizacional no lugar para lhe fornecer respaldo.
- *Dados empresariais e estratégia de SOA.* A SoBI baseia-se na existência de um plano estratégico para os aplicativos de SO na organização e no reconhecimento da importância do sistema de armazenamento dos dados de registro em armazenamentos ou aplicativos de nível empresarial. Por exemplo, a organização não armazena as principais informações comerciais nas planilhas.
- *Relatórios operacionais versus administrativos.* Deve-se fazer uma clara delineação entre os tipos operacionais e administrativos de relatórios que serão necessários na estrutura SoBI. A SoBI BI será a versão única da verdade para os relatórios administrativos. Terá o alcance suficiente para atender às exigências específicas da empresa em termos de inteligência comercial. O armazém de dados de suporte conterá apenas os dados necessários para suportar essas exigências.
- *Propriedade dos dados.* Embora o armazém de dados contenha os dados coletados de múltiplas fontes e sistemas de dados, de uma perspectiva orientada ao serviço, o armazém de dados não deve ser considerado a versão principal ou o armazenamento-padrão de todas as exigências de dados. O proprietário desses dados permanece sendo o sistema de registro.

## Fatores de sucesso

Vamos dar uma olhada nesses fatores de sucesso em mais detalhes. A governança é um fator de sucesso fundamental no desenvolvimento de uma solução orientada ao serviço bem-sucedida. É importante que uma organização possa controlar e anunciar os serviços, os formatos de mensagens e as estruturas que são suportados para impedir uma proliferação não controlada dos serviços, das mensagens e das definições de entidades no ambiente. Isso está intimamente ligado às atividades de definição de esquema e exige um gerenciamento ativo por parte de um corpo de governança para garantir que o sistema adere aos princípios orientados ao serviço.

A chave é encontrar um equilíbrio no nível da governança e ter controle suficiente para fornecer uma estrutura para o desenvolvimento e a implantação bem-sucedidos dos serviços, mas não ao ponto de debilitar a capacidade do sistema de responder, de forma ágil, às necessidades da empresa.

Em qualquer projeto de BI, a falta de clareza na função do armazém de dados e no escopo dos dados neles contidos pode levar a problemas. Em um exercício de projeto duradouro, se o escopo do armazém de dados não for gerenciado, o exercício de projeto cresce à medida que são descobertas fontes de dados em potencial, e os dados nessas fontes precisam ser consolidados no modelo de armazém de dados.

O modelo de armazém de dados precisa ser projetado para garantir a capacidade de expansão, a fim de garantir que os dados de ativos diferentes podem ser adicionados à medida que se possa fazer um caso de negócio para os dados. Não é algo prático partir do princípio de que os dados de todos os aplicativos no cenário organizacional possam ser incluídos no projeto do primeiro modelo de dados. É mais provável que uma abordagem incremental do projeto e do desenvolvimento do armazém seja mais bem-sucedida.

Para a estratégia dos dados e da SOA empresarial, a aplicação bem-sucedida da SoBI baseia-se na existência de um plano estratégico organizacional para os aplicativos de SO e para o sistema dos dados de registro a serem mantidos nos armazenamentos ou nos aplicativos empresariais. Quando os sistemas e os dados não conseguem suportar a integração direta na SOA proposta pela estrutura SoBI, parte-se do pressuposto de que o plano eventual da empresa é migrar esses sistemas e fontes de dados para uma plataforma que suportaria esse nível de integração. Entre os exemplos dos tipos de sistemas

mencionados nessa pressuposição estão os sistemas que são mais carregados e não podem lidar com o fardo extra de expor os serviços e sistemas que não suportam pesquisas online e se baseiam em exportações periódicas em lotes.

Algumas das informações essenciais ao negócio são mantidas em armazenamentos ou aplicativos que não são apropriados aos dados do valor desse negócio – por exemplo, as planilhas do Microsoft Excel que guardam versões principais dos dados tornando a planilha o sistema de registro. Deve haver uma direção estratégica que leve ao ponto no qual os dados são mantidos em um armazenamento ou aplicativo que tenham força em toda a empresa, e para os documentos se tornarem visualizações desses dados, e não as fontes principais. O objetivo é transformar os documentos enquanto sistemas de registro em documentos enquanto visualizações dos dados mantidos nos sistemas de registro.

## Manter a integridade

Deve-se fazer uma clara delineação entre os tipos operacionais e administrativos de relatórios que serão necessários na estrutura SoBI. A visão tradicional do armazém de dados como fonte única de todas as necessidades de informações corporativas não se mantém na estrutura SoBI. O objetivo da estrutura SoBI é apoiar o que há de mais importante na BI como "a versão única da verdade", mas também manter a integridade dos sistemas de registro como proprietários dos dados corporativos.

É provável que um relatório operacional atenda a um destes critérios: exigir acesso ao vivo aos dados; ser necessário para o gerenciamento operacional da empresa (por exemplo, as informações numa transação individual); não precisar de dados históricos para comparações; e não precisar de dados resumidos (dados agregados, exceto para o total básico). Em geral, define-se um relatório administrativo como algo que:

- Exige um acesso não imediato aos dados, mas na hora certa, e em geral exige que os dados resumidos sejam apresentados em um cronograma histórico predeterminado (por exemplo, comparações métricas mês a mês por ativo)
- Apresenta ao usuário a capacidade de explorar os dados apresentados de acordo com a sua vontade, para pesquisar rapidamente os problemas em potencial de desempenho (por exemplo, aumentar o detalhamento dos dados). O relatório pode destacar as áreas de falha com base na formatação condicional.
- É definido apenas para avisar o usuário quando ocorrer um problema (relatório de exceção).
- Auxilia na previsão do desempenho comercial.
- Ajuda nas metas subjacentes da pessoa e da empresa (por exemplo, a eficiência de produção, a sobrevenda e a maximização do lucro do produto).

Embora o armazém de dados contenha os dados coletados de múltiplas fontes e sistemas de dados, de uma perspectiva orientada ao serviço, o armazém de dados não deve ser considerado a versão principal. O proprietário desses dados permanece sendo o sistema de registro.

A SO discrimina claramente entre dois tipos diferentes de dados, a saber, os dados que são mantidos no serviço e os dados que o serviço expõe aos aplicativos ou a outros serviços. Os dados internos são aqueles que o serviço utiliza para realizar as operações que ele fornece. Essas informações são inteiramente privadas ao serviço, nunca sendo expostas diretamente. Os dados externos são aqueles publicados pelo serviço e usados para trocar informações e solicitações com os clientes do serviço. Esses dados são definidos explicitamente em termos de um esquema organizacional.

O aplicativo que é proprietário dos dados (também conhecido como sistema de registro) é responsável, no fim, por manter seus próprios dados; portanto, para permitir que outros aplicativos solicitem mudanças nos dados, o aplicativo proprietário precisa expor a funcionalidade de atualização de solicitação por meio de sua interface de serviço. (Consulte Recursos para obter mais informações sobre esses dois tipos de dados.)

Ao trabalhar com a integração de dados, a SoBI será flexível o suficiente para trabalhar com esses cenários de integração principais.

- Volumes de dados, que são transações exclusivas ad hoc ou cargas de dados em massa de grande volume.
- Integração com aplicativos empacotados e esquemas de bancos de dados "proprietários" de terceiros.
- Consolidação do banco de dados.
- Integração de fontes de dados relacionais, não relacionais, estruturadas e semi-estruturadas e de sistemas legados.
- Suporte a serviços da Web e integração com middleware de troca de mensagens.

Ao trabalhar com a BI, a SoBI visará a satisfazer várias exigências. As empresas precisam coletar e agregar informações de fontes diferentes dentro da organização e conseguir compartilhar essas informações de uma maneira aberta, com um estado diverso de aplicativos em partes diferentes da organização, sem primeiro conhecer em detalhes como as informações serão utilizadas. As empresas também precisam isolar os usuários do formato e da estrutura subjacentes dos dados, em vez de se focar no significado dos dados de negócio dentro da organização. Os consumidores das informações estão preocupados principalmente com a semântica dos dados, e não com a sintaxe deles. Partes diferentes da empresa precisam conseguir compartilhar uma linguagem comum quando descrevem a si próprias, e as empresas precisam publicar os dados de forma mais ampla dentro da organização, diferenciando a publicação dos dados e os dados para análise.

A publicação de informações definidas, tais como KPIs, ou métricas por meio de mecanismos abertos como os serviços da Web, permite que essas informações sejam consumidas facilmente na organização, sem o recurso dos aplicativos especializados. A publicação de dados para análise ainda será uma parte principal dos serviços prestados pelo armazém de dados, mas isso tende a se focar em um público mais limitado na organização, com o acesso aos aplicativos especializados necessários para a análise dos dados. Uma das metas da SoBI é permitir a ampla divulgação das informações a um público mais amplo de usuários, aplicativos e outros serviços na organização.

## Padrões de integração da SoBI

Outro objetivo da estrutura SoBI é ter uma abordagem pragmática do trabalho com sistemas que não podem ser integrados diretamente na arquitetura – por exemplo, os que não podem suportar carga extra ou aqueles que não são dimensionáveis o suficiente para suportar a integração direta.

Tendo isso em mente, a estrutura define um conjunto de cenários (ou padrões) que descrevem categorias de sistemas de fonte típicos que os autores encontraram, junto com um esboço recomendado para integrar cada categoria do sistema. Prevê-se que essa coleção de padrões de integração evoluirá e se desenvolverá à medida que sistemas mais diversos são integrados em uma solução SoBI. Esses padrões ajudam a resolver uma das principais prioridades orientadas a serviço, qual seja, a capacidade de conseguir suportar a substituição de aplicativos com impacto mínimo na empresa. A estrutura SoBI descreve como esses sistemas podem ser abstraídos de uma forma que garanta que haja uma interrupção mínima quando os sistemas são finalmente substituídos.

Até agora conseguimos estruturar a SoBI em termos de um cenário mundial ideal no qual os vários sistemas de registro podem participar da

**Figura 12** Abordagem não adequada à interface do serviço

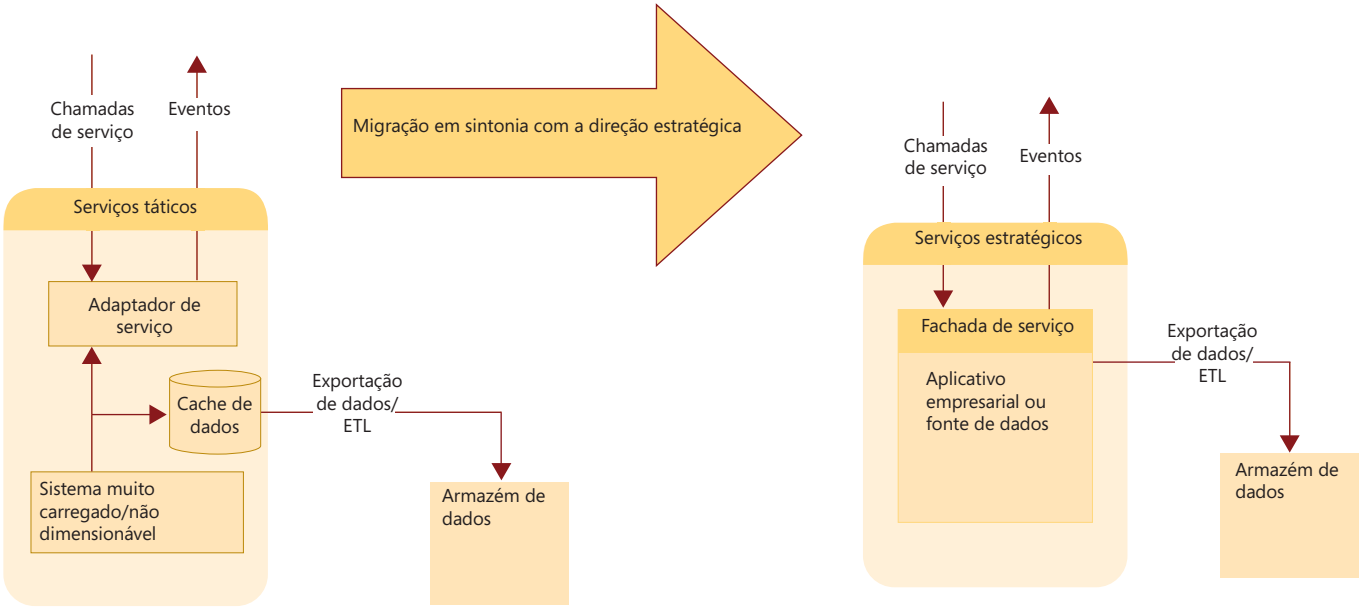
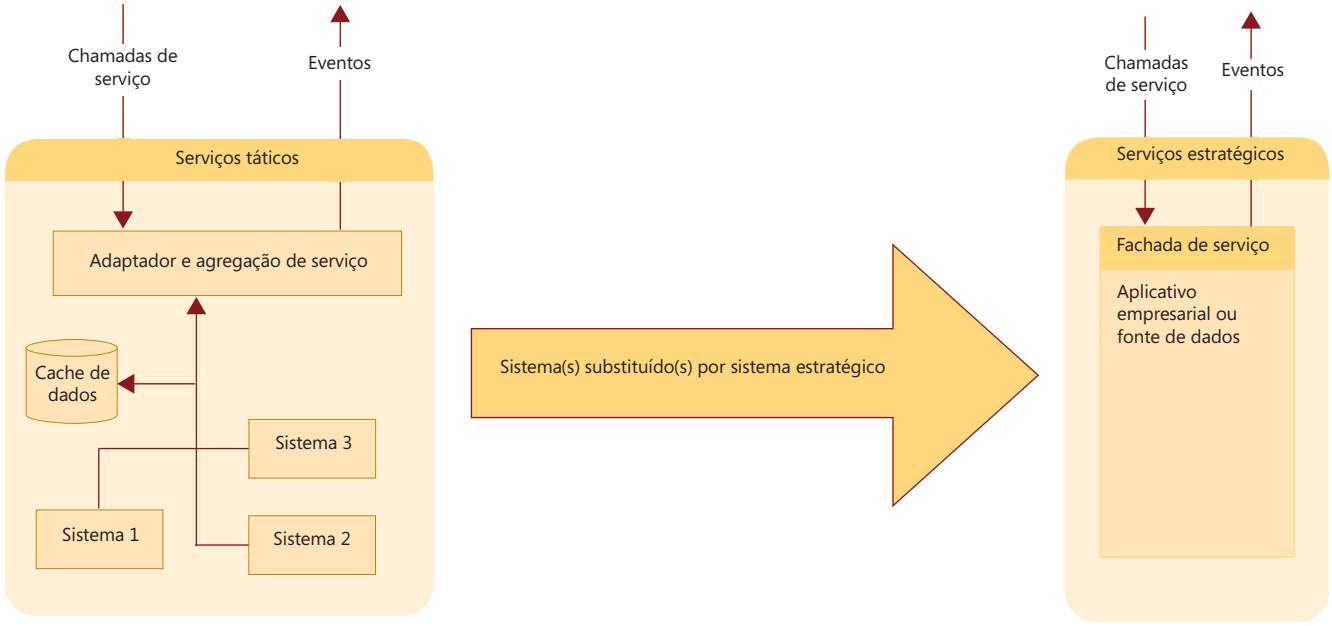



**Figura13** Substituição da implementação de BI




arquitetura SoBI de uma maneira orientada ao serviço. Em um ambiente de projeto real, é provável que haja uma série de restrições que influenciam a nossa capacidade de implementar a SoBI pura. Identificamos várias categorias de restrições, que serão discutidas brevemente, e como exemplos em cada caso identificamos um padrão, que, ao ser utilizado em conjunto com o princípio da SoBI na SOA empresarial e na estratégia de dados, garante que a implementação pragmática da SoBI permanece dentro dos princípios orientadores da estrutura SoBI.

## SoBI pura

Uma situação ideal é aquela em que o aplicativo é dimensionável e flexível o suficiente para suportar a exposição dos serviços e eventos ao barramento de serviço, ou diretamente ou por meio de uma fina fachada de serviço (consulte a Figura 8). Essa categoria de aplicativo também é capaz de suportar a exportação dos dados para o armazém de dados, conforme necessário.

Uma das restrições no tipo do sistema de fonte é o processamento transacional/em tempo real. Alguns aplicativos conterão as informações que são interessantes da perspectiva da análise e de uma perspectiva em tempo real. Tais aplicativos podem ser habilitados para o serviço criando-se uma fachada de serviço que expõe os serviços do aplicativo à organização, e que dispara eventos quando são realizadas mudanças ou atualizações "interessantes". Nesse cenário, os aplicativos que estão interessados nos eventos desse aplicativo podem se subscrever para os eventos publicados. Os assinantes incluirão um agente do armazém de dados que ordena os eventos para a integração com o armazém de

dados (consulte a Figura 9).

Os sistemas não estruturados e semi-estruturados são outra restrição. Em qualquer ambiente complexo, provavelmente haverá uma série de fontes de dados que contêm informações que devem ser consumidas pela solução, mas que são mantidas em formatos semi-estruturados ou não estruturados, tais como sistemas de gerenciamento de planilhas e documentos. Com esses tipos de sistemas, é importante estruturar as informações antes da integração no armazém de dados, e será possível fazer isso em uma de várias formas. Uma delas é aplicar a estrutura ao armazenamento de dados. Por exemplo, a provisão de modelos estruturados e controlados por mudanças para as planilhas e documentos, de tal forma que as informações possam ser extraídas com precisão e confiabilidade do documento, envolverá a mudança do processo comercial. Outra forma é impor a estrutura na leitura dos dados realizada no armazenamento, que é algo inerentemente difícil na medida em que se baseia em pressuposições sobre a semântica da estrutura existente da fonte de dados e na confiança de que isso nunca muda. Se essa pressuposição se mantiver, é possível realizar uma extração pragmática.

Estamos partindo do pressuposto de que as fontes de dados não empresariais serão por fim atualizadas para suportar mais diretamente os serviços que elas fornecem (consulte a Figura 10). É importante mudar os documentos das fontes de informações para visualizações das informações (consulte a Figura 11).

Agora vamos dar uma olhada nas restrições de limitação do sistema de fontes. Para os sistemas muito carregados, haverá ocasiões em que há fontes de dados ou sistemas que contêm dados que não podem ser interrogados em tempo real devido a restrições tecnológicas ou operacionais. Considere estes exemplos: um sistema muito utilizado talvez não consiga suportar o acréscimo de uma interface de serviço que processa um novo grupo de pesquisas em uma base freqüente; uma fonte de informações que não suporta o acesso simultâneo, como os dados mantidos em uma planilha; e o acesso ad hoc ou em tempo real aos dados em um sistema de produção é considerado um risco para a execução diária eficaz do sistema essencial à empresa.

Nesses cenários, a solução tática é fazer cache dos dados em um sistema que pode então fornecer uma interface definida e publicada para os dados ou serviços. Esse cache permite que os aplicativos e serviços tenham acesso às informações mais atualizadas possíveis por meio de um serviço exposto. Essa abordagem fornece uma maneira de dimensionar uma fonte de dados ou aplicativo para atender às exigências do negócio de uma forma que garante um claro desligamento dos dados ou do aplicativo e do serviço prestado. Esse desligamento é importante para o desenvolvimento futuro, quando o aplicativo ou os dados podem ser transferidos para uma solução mais dimensionável e prestar o serviço diretamente, ou quando o aplicativo pode ser aprimorado para suportar o serviço diretamente. Uma advertência que se pode fazer aqui é que haverá ocasiões em que o tipo de dado que é necessário pelo sistema é puramente de natureza da inteligência comercial, tais como a exportação de dados de larga escala. Nesses cenários, a abordagem da interface de serviço não será adequada (consulte a Figura 12).

Estamos partindo do pressuposto de que os sistemas muito carregados serão por fim atualizados para suportar mais diretamente os serviços que eles fornecem.

## Baixa expectativa de vida

Um dos princípios da SoBI é que a organização deve colocar em vigor um plano estratégico para os aplicativos orientados ao serviço e para o sistema de dados de registro a ser mantido nos armazenamentos ou aplicativos da empresa. Esse plano significará inevitavelmente a substituição de alguns sistemas de registro no atual cenário, e em alguns casos é bem provável que sejam substituídos após a implementação do projeto de BI. Portanto, é fundamental que seja definida uma abordagem que possa suportar esses sistemas nos anos finais de suas vidas e facilitar o processo de substituição quando o próximo sistema ganhar vida (consulte a Figura 13). Essa abordagem vai precisar produzir uma arquitetura que garanta o equilíbrio entre uma transição fácil para a nova fonte de dados e o compromisso de empenhar a equipe do projeto em um grande esforço de desenvolvimento que no final será jogado fora.•

### Recursos

"Data on the Outside vs. Data on the Inside", Microsoft Developer Network (MSDN) Whitepaper, Pat Helland, Microsoft Corporation http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/dataoutsideinside.asp

"Data Warehousing Lessons Learned: Trends in Data Quality", Lou Agosta, coluna publicada em *DM Review Magazine* (February 2005)

"Information as a Service: Service-Oriented Information Integration", Ronald Schmelzer, Zapthink www.zapthink.com/report.html?id=WP-0125

"Information Bridge Framework: Bringing SOA to the Desktop in Office Applications", Ricard Roma i Dalfo, *The Architecture Journal 4*, (Microsoft Corporation, 2004) http://msdn.microsoft.com/architecture/default.aspx?pull=/library/enus/dnmaj/html/ibf-J4.asp

"Over half of data warehouse projects doomed", Robert Jaques, Gartner, (February 2005)

"Service Orientation and Its Role in Your Connected Systems Strategy", artigo que fornece uma visão geral da visão da Microsoft sobre a orientação ao serviço e a arquitetura orientada ao serviço na computação empresarial. (Microsoft, 2005) http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnbda/html/srorientwp.asp

"Service-Oriented Architecture: Considerations for Agile Systems", Lawrence Wilkes and Richard Veryard, CBDI Forum, The Architecture Journal 2 (Microsoft, 2004) http://msdn.microsoft.com/architecture/journal/default.aspx?pull=/library/en-us/dnmaj/html/aj2service.asp

"Service-Oriented Architecture: Implementation Challenges", Easwaran G. Nadhan, Principal, EDS The Architecture Journal 2, (Microsoft, 2004) http://msdn.microsoft.com/architecture/journal/default.aspx?pull=/library/en-us/dnmaj/html/aj2soaimpc.asp

### Sobre os autores

**Sean Gordon** é arquiteto de estratégia de sistemas empresariais e de prática de consultoria em arquitetura da Microsoft e trabalha no escritório da empresa na Escócia.

**Robert Grigg** é consultor administrativo e arquiteto empresarial em Conchango, no Reino Unido.

**Michael Horne** é consultor administrativo e arquiteto de inteligência de negócio em Conchango, Reino Unido.

**Simon Thurman** é arquiteto no grupo de desenvolvimento da Microsoft Developer Network, Reino Unido.

# Planejamento da arquitetura técnica

por Waleed S. Nema


## Resumo

A necessidade de haver um planejamento da arquitetura técnica é bem conhecida por administradores, auditores e equipe técnica nesta era da informação altamente exigente e em constante transformação. Os impulsionadores comerciais dessa necessidade incluem o alinhamento da TI com a empresa e o controle de níveis e gastos com serviços. Essa discussão resume a primeira experiência de criar um plano tático de infra-estrutura de dois anos com a WSA (Windows Server Architecture) para um departamento de operações de computador. A arquitetura técnica é definida junto com a cobertura dos produtos entregues e dos desafios envolvidos.


A disciplina de planejar uma arquitetura técnica se tornou mais conhecida apenas recentemente. Portanto, ela não é muito bem compreendida devido à falta de experiência, de treinamento e até mesmo de literatura. O planejamento da arquitetura técnica, quando feito corretamente, tenta abrir todos os aspectos do ambiente atual e mapeá-los em um estado desejado e consistente com a empresa. Ele tende a criar uma ampla variedade de dúvidas e problemas, alguns sobre os quais nunca se pensou antes: políticos, comerciais e outros. À medida que você tenta esclarecer a visão e o escopo do planejamento da arquitetura, encontra resistência devido a entendimentos ambíguos ou agendas ocultas. Você pode até ter alguma resistência ao saber quem você é ou de onde está vindo.

Partir do pressuposto de que a visão e o escopo do planejamento de arquitetura estão de acordo não é uma garantia para uma participação forte, aberta e integral na equipe de operações. As vantagens de divulgar informações devem pesar mais do que o custo de revelar pontos fracos e vulnerabilidades dos quais apenas a equipe operacional tem conhecimento. Depois que o ciclo de planejamento está concluído e se comprova que a participação é recompensada mediante a alocação de orçamentos e recursos é que você pode esperar mais participação nos ciclos subseqüentes. Estando em desvantagem no primeiro, você precisará fazer com que a mensagem seja ouvida por todos, e que a administração a transmita com clareza. Para ser bem-sucedida, a administração deve destacar claramente as vantagens do planejamento de arquitetura desde o início.

Como pode ser visto, você tem uma divisão justa de desafios já ao iniciar o processo de planejamento de arquitetura, mais desafios ao manter a participação da equipe operacional e outros desafios ao obter o suporte da administração. E, se seguir o modelo de supervisão dos planos de ação de acompanhamento, como descrito posteriormente, você poderá parecer um policial ou um auditor.

A única maneira de esse esforço ser bem-sucedido é mostrar e provar as boas intenções. A administração, que geralmente fornece a diretiva para essa atividade, junto com a equipe de arquitetura, deve convencer a equipe que essa atividade não tem a ver com exposição, mas com a criação de um ambiente operacional melhor e mais eficiente que melhor atenda aos interesses da empresa. No final, todos saem ganhando, mas à custa de aceitar as mudanças e eliminar as defesas e os obstáculos.

## Dando vida

Acima de tudo, os membros da equipe de arquitetura são facilitadores. Eles devem trabalhar com a equipe operacional para compreender o ambiente atual. Na realidade, a equipe operacional deve ser a arquiteta do ambiente atual, pois o conhece melhor do que ninguém. A equipe de arquitetura deve trabalhar com a empresa e com outros planejadores corporativos para incorporar a estratégia e a direção do negócio com as quais a TI deve entrar em sintonia. A equipe deve trabalhar com a administração para compreender as táticas e as restrições e conseguir suporte. Ela também deve incorporar a perspectiva das práticas recomendadas da indústria, englobando estrutura, tecnologia, processos e pessoas. A equipe de arquitetura precisa reunir tudo isso em um plano realista e exeqüível.

A arquitetura e o planejamento geralmente fornecem uma visão de um estado futuro que traduz algum tipo de necessidade ou desejo. Se estamos falando de uma casa, por exemplo, o arquiteto desenvolve um plano detalhado que atende às orientações do proprietário. Se estamos falando de um modelo a ser usado em uma comunidade de

"PARTIR DO PRESSUPOSTO DE QUE A VISÃO E O ESCOPO DO PLANEJAMENTO DE ARQUITETURA ESTÃO DE ACORDO NÃO É UMA GARANTIA PARA UMA PARTICIPAÇÃO FORTE, ABERTA E INTEGRAL NA EQUIPE DE OPERAÇÕES"

desenvolvimento, então estamos falando de um padrão a ser seguido para manter um orçamento específico e uma aparência geral. As orientações e as exigências do proprietário neste sentido também poderiam ser pensadas como um padrão para o construtor. Para dar vida a um projeto arquitetônico, é necessário projetar e seguir um projeto de construção bastante complexo. A beleza da casa no papel não significa nada até que o proprietário possa vê-la ao vivo e a cores.

Dessa forma, faz sentido pensar no planejamento arquitetônico como um processo que converte um conjunto de orientações e exigências em padrões que os construtores possam implementar. Se já existe uma construção, a arquitetura significa projetar novas exigências e aprimoramentos na estrutura existente.

O mundo técnico ainda não aprendeu o suficiente com a indústria bem-estabelecida da construção civil. As funções do proprietário, do arquiteto, do construtor e do supervisor são bem-conhecidas na obra, mas não na área técnica, particularmente a função do inspetor. Na indústria da construção civil, o construtor quase sempre é uma entidade diferente do arquiteto. O supervisor também poderia ser uma entidade independente, mas freqüentemente é igual ao arquiteto, e ambos agem em prol do proprietário e devem aprovar o trabalho do construtor.

Nós da área técnica precisamos delegar autoridade aos arquitetos e permitir que eles assumam a função de inspecionar e supervisionar a implementação dos planos de arquitetura. Na TI, os patrocinadores do projeto geralmente são os proprietários. Os arquitetos devem agir em nome deles e assumir a responsabilidade sempre que for necessária autoridade. Com essa nova visão do processo, o planejamento arquitetônico não apenas é responsável por produzir plantas e padrões, como também deve ser responsável por supervisionar a implementação dos planos de arquitetura.

**"COMO OS ARQUITETOS REPRESENTAM OS PATROCINADORES EXECUTIVOS NESTE MODELO, ELES PRECISAM ESTAR CONSCIENTES DO NEGÓCIO E DEVEM IDENTIFICAR AS RECOMENDAÇÕES DE AÇÕES E DEFINIR AS PRIORIDADES DE ACORDO COM ISSO"**

Como os arquitetos representam os patrocinadores executivos neste modelo, eles precisam estar conscientes do caso de negócio e devem identificar as recomendações de ações e definir as prioridades de acordo com isso. O patrocinador executivo é, no final das contas, responsável pelo processo arquitetônico e por sua implementação bem-sucedida.

Para resumir, o planejamento da arquitetura deve dar conta destes cinco aspectos: padronização, aprimoramentos, supervisão de implementação, prioridades orientadas a negócios e acompanhamento da administração; e podemos defini-lo desta forma: o planejamento da arquitetura técnica é um processo patrocinado pela administração executiva que converte as atuais necessidades comerciais, os desafios e os desejos em um conjunto priorizado de padrões, planos e itens de ação, que recomendam e supervisionam os aprimoramentos de serviço levando a uma melhor experiência do cliente e à excelência operacional.

### O que é e o que não é

A arquitetura técnica visa a edificar a capacidade e a maturidade do serviço. Tem a ver com modelos, padrões e ações (aprimoramentos). Trata-se de uma especificação de solução de alto nível, mas não é uma especificação de projeto. As especificações de solução são como definir o orçamento e o tamanho do terreno no projeto de uma casa em uma comunidade de desenvolvimento, sem entrar em suas especificações de layout. Depois de aprovados, os projetos de implementação e acompanhamento são geralmente sustentados para novos serviços ou reformas de grande escala.

A arquitetura técnica não é simplesmente um diagrama do Visio de configurações do servidor de arquivos. A arquitetura é, não importando se é utilizado um espaço de nome lógico no servidor de arquivo (como o Sistemas de arquivos distribuídos), se é seguido um modelo de gerenciamento centralizado ou distribuído; ou se é utilizado um modelo de hospedagem compartilhada de acesso livre para os bancos de dados ou para os sites na Internet. Esses tipos de decisões têm conseqüências de longo alcance e, portanto, são decisões de nível de arquitetura.

## Projeto tático para o Windows Arquitetura de servidor

Esse projeto é uma tabela parcial do conteúdo do plano tático da WSA.

- Introdução
- Disciplinas
  - Serviços de hospedagem na Web
    - Resumo executivo
    - Introdução
      - Informações importantes
      - Impulsionadores comerciais
      - Objetivos
      - Escopo
    - Conceito de solução
      - Arquitetura de serviço
        - Quadrante de Otimização do MOF
          - Modelo de serviço
          - Ofertas de serviço
          - Compromissos de nível operacional/de serviço
          - Solicitações de Serviço
          - Garantia de qualidade
          - Modelo de equipe
          - Serviços de suporte
      - Arquitetura de tecnologia
        - Geral
          - Estratégia corporativa de aplicativos
          - Definições
          - Zonas de confiança
          - Perfis de hospedagem (padrões de implantação)
          - Orientações arquitetônicas
        - Quadrante de Mudança do MOF
          - Gerenciamento de dependências
          - Controle de produção
        - Quadrante de Operação do MOF
          - Monitoramento
        - Quadrante de Suporte do MOF
          - Gerenciamento de incidentes/problemas
          - Isolamento do problema
          - Relatórios aprimorados de erros
        - Quadrante de Otimização do MOF
          - Estratégias de segurança
          - Estratégias de continuidade comercial
    - Recomendações do plano de ação
    - Apêndice
      - Referências
      - Diagramas
  - Outros serviços (arquivo e impressão, diretório, dados)
- Esquema do servidor
- Consolidação dos produtos entregues
  - Infra-estrutura
  - Desenvolvimento
  - Políticas e processos

**Tabela 1 Exemplo de recomendações do plano de ação**

| Seção | Prioridade/ prazo | Objetivo | Recursos/ habilidades | Estimativa de trabalho | Métrica de sucesso |
|---|---|---|---|---|---|
| 3.x | Modelo de serviço | | | | |
| | Prioridade A T4 2005 | Divulgar o modelo de hospedagem do ISP (Provedor de Serviços de Internet) | WHG | | Publicar documentos em SLC, OLC e os novos termos e condições de solicitação de serviço |
| 3.y | Monitoramento | | | | |
| | Prioridade A T2 2006 | Monitorar sites/ servidores em relação à inatividade, ao desempenho e ao otimização | WHG | | Atribuir uma função de monitoramento a uma pessoa conhecida que será responsável por tratar de todos os eventos do console de monitoramento. |
| | | | | | Explorar os novos recursos do MOM 2005, inclusive o IIS Management Pack, e os sites e MP de serviços. |
| | | | | | Habilitar as mensagens de texto e o aumento para os erros críticos, tais como quando o servidor sai do ar, por exemplo. |
| 3.z | Controle de produção | | | | |
| | Prioridade B T4 2006 | Desenvolver uma ferramenta de publicação de um site | Atribuição de consultoria | | Firmar um contrato de desenvolvimento que construirá uma Ferramenta de Publicação de sites, como definido no protótipo existente. |

A Arquitetura técnica tem a ver com processos, com certeza. Na realidade, é freqüentemente o processo, e não a tecnologia, que pode fazer ou arruinar um serviço. Os processos geralmente envolvem problemas com as pessoas, os quais implicam questões políticas e, às vezes, são o aspecto mais perigoso de todos. Pior ainda é quando o sistema não conta com um sistema de limitações e inspeções na hora em que se trabalha com esses problemas.

A arquitetura técnica tem a ver com medidas. As pessoas devem sentir que as medidas são criadas para ajudá-las, e não para expô-las. A administração deve tornar suas intenções tão claras quanto o cristal, para ganhar a confiança e o total empenho da equipe. Somente então vamos ter a chance de fazer as coisas funcionarem. A administração deve transmitir as razões para a existência dessas medidas, que incluem a percepção, ou a consciência, do quão distante estamos, o aprimoramento para atingir as metas e a avaliação do que é necessário. A administração também deve evitar o constrangimento público colocando um tempero positivo nos relatórios públicos, tais como o "aumento das percentagens no último período", mantendo os relatórios absolutos na esfera interna da empresa até que os números estejam próximos às metas do KPI (Indicador Principal de Desempenho). A administração deve provar as boas intenções obtendo os recursos necessários ou aceitando um progresso mais lento e resultados de menor impacto.

**"A ARQUITETURA É UM CICLO CONTÍNUO COMO ASSEGURADO POR MUITAS ESTRUTURAS, COMO A MICROSOFT OPERATIONS FRAMEWORK, O CICLO DE DEMING E O CICLO COBIT/ADMINISTRAÇÃO"**

A arquitetura técnica tem a ver com automatização. Essa é a chave para a eficiência, para os processos determinantes e para a estabilidade. A automatização dá à equipe de operações mais tempo para se focar na análise e na otimização, que são a base para o aprimoramento do serviço.

A arquitetura técnica também tem a ver com monitoramento. A única maneira de estar no controle é olhar constantemente para a estrada e para as placas de sinalização. A arquitetura fornece um esquema e garante que você vai ver os avisos certos. Também o ajuda a criar um sistema de alertas e dimensionamentos – possivelmente uma reação corretiva automatizada.

A arquitetura é um ciclo contínuo como assegurado por muitas estruturas, como a Microsoft Operations Framework, o ciclo de Deming e o ciclo COBIT/administração. É basicamente uma fase de planejamento seguida por operações monitoradas, cuja aprendizagem promove a otimização e os aprimoramentos em um novo ciclo.

## Documentando o plano

O corpo principal do plano tático da WSA encontra-se nas disciplinas da área funcional, como os serviços de hospedagem da Web, os serviços de diretório e assim por diante (veja o texto no quadro, "Plano tático para a Windows Server Architecture"). O esquema do servidor se aplica a mudanças específicas de tecnologia no período tático. A última seção, produtos entregues de consolidação, coleta as recomendações do plano de ação de todas as seções e os categoriza em três áreas para ajudar nos projetos de acompanhamento: infra-estrutura, desenvolvimento, políticas e processos.

O corpo principal da arquitetura do serviço de cada disciplina – os serviços de hospedagem da Web, os serviços de diretório e assim por diante – encontra-se na seção de conceito de solução, que é dividido nas arquiteturas de serviço e tecnologia as quais são, por sua vez, baseadas na MOF (Microsoft Operations Framework). A introdução inclui seções de contexto, impulsionadores comerciais, objetivos e escopo, e estabelece objetivos como o número de 9s na alta disponibilidade, entre outros itens.

O resumo executivo tem uma página de extensão e é fornecido "no estado em que está", demonstrando os benefícios em potencial, as metas e a abordagem, além das recomendações mais significativas do plano de ação.

As recomendações do plano de ação são, provavelmente, o resumo mais importante de todas as especificações de solução e métricas de sucesso, pois a equipe operacional estabelece as prioridades e os prazos e indica na coluna recursos/habilidades se eles mesmos podem fazer o trabalho ou se precisam terceirizar o serviço (consulte a Tabela 1). Estabelecer o planejamento da arquitetura como disciplina e função é algo que compensa mesmo antes de encerrar o primeiro ciclo tático. É uma alegria muito grande ver a conclusão do plano e o início da ação. Esse resultado não acontece sem custo ou esforço e depende do trabalho em equipe entre todos os participantes. •

**Sobre o autor**

**Waleed Nema** é líder de equipe no planejamento da arquitetura técnica do Windows no departamento de operações de computadores, na Saudi Aramco. Pelo grande sucesso deste projeto, Waleed gostaria de agradecer a administração da empresa, a equipe operacional e o grupo de arquitetos.

# Behavioral Software Architecture Language

por Behzad Karim


## Resumo

A arquitetura de software é uma palavra-chave muito importante atualmente para as pessoas da nossa profissão. Parece que todo mundo está buscando o verdadeiro potencial da arquitetura de softwares e o que ela pode fazer pelas pessoas. A idéia básica da definição de arquitetura é projetar uma estrutura de softwares e uma interação de objetos antes da fase detalhada do projeto. Embora uma definição séria de arquitetura esteja sendo sugerida apenas pelos grandes projetos, qualquer trabalho de construção ou implementação de software deve ser precedido por uma fase de projeto e aprovação de arquitetura. Conheça a BASL, uma linguagem que integra a definição de arquitetura de software à implementação do software (codificação).


É bem possível que alguém argumente que há pelo menos uma dezena de padrões e ferramentas bem definidos que podem ser usados para definir a arquitetura de software. Realmente temos padrões e ferramentas para definir, divulgar e até gerar modelos para o projeto de softwares. Algumas dessas ferramentas podem até funcionar de duas formas na hora de converter o código para um modelo de arquitetura e vice-versa. Por que então sente-se uma necessidade de outra linguagem ou modelo de programação?

Embora haja inúmeras formas de definir a arquitetura de software em termos de pacotes, componentes e conectores (ou seja, a estrutura do software), quando se trata de definir o comportamento dinâmico dos softwares, somos incapazes de fornecer uma definição, divulgá-lo ou até mesmo projetá-lo com clareza. No entanto, o comportamento dinâmico dos softwares é realmente o que eles fazem depois que são enviados para o ambiente de produção.

A maioria das técnicas atuais e bem-conhecidas (e estabelecidas) utilizadas para a definição da arquitetura de softwares vem de uma era em que essa arquitetura ainda era um mito. Embora essas técnicas (e ferramentas) façam um ótimo trabalho na hora de definir a estrutura e os componentes dos sistemas de softwares, elas carecem da capacidade de encapsular e definir o comportamento dos softwares e as várias interações com o ambiente, em relação à dimensão do tempo. Indiscutivelmente essas ferramentas não foram feitas para o arquiteto de software, como o foram para o engenheiro de software.

A necessidade básica do arquiteto de definir a essência de um sistema de software de uma maneira abstrata, enquanto fornece a imagem do sistema ativo, permanece sem resposta. Ou estamos indo muito fundo nos detalhes da implementação ou estamos meramente transmitindo a superfície mais externa do sistema. Em ambos os casos, estamos deixando de fora o ingrediente mais importante do caráter do software: os aspectos comportamentais dinâmicos do sistema. Em muitos projetos, esses aspectos do sistema estão sendo descobertos no projeto detalhado ou na fase de codificação pela equipe de desenvolvimento.

## Encarando a realidade

Outro dilema é a lacuna cada vez maior entre o código e a arquitetura original do sistema. Estamos usando ferramentas e linguagens diferentes no processo de construção do software. O arquiteto de software está usando algum tipo de linguagem de modelagem, ao passo que o desenvolvedor está usando uma linguagem de programação. Conseqüentemente, em especial depois que o sistema passa para a fase de produção, os documentos, os diagramas e o código saem de sincronia. Será que você já teve que voltar e revisar um sistema que você projetou no passado, e aí acabar descobrindo que a arquitetura não tinha nenhuma semelhança com o projeto original?

Essa analogia mostra-se verdadeira para os diagramas de classes, os casos de uso e os cenários de testes. O problema usual com esses documentos é que todos eles parecem se tornar obsoletos depois que o software vai para produção. Tais problemas surgem com a falta de um meio compartilhado ou um ponto singular de referência para a arquitetura e a solução física. Em teoria, a arquitetura e o código são faces inseparáveis da mesma realidade: o sistema de software.

> **"O RESULTADO FINAL DO NOSSO TRABALHO NÃO É MERAMENTE UM PRODUTO OU COMMODITY ESTÁTICO, MAS UM *ORGANISMO VIVO, QUE RESPONDE AOS ESTÍMULOS*"**

A disciplina de engenharia de software já percorreu um longo caminho desde seus primeiros dias de existência. Ao longo da jornada, as idéias de outras profissões ligadas à engenharia foram utilizadas. Um fator fundamental que diferencia nossa profissão de outros campos da engenharia é a natureza do nosso trabalho e, mais especificamente, o resultado do nosso trabalho. Começamos com pensamentos, idéias e visões básicas e as desenvolvemos em realidades que podem mudar a vida de milhões de pessoas. O resultado final da nossa mão-de-obra não é meramente um produto ou commodity estático, mas um *organismo vivo, que responde aos estímulos.*

A engenharia de software enquanto disciplina continuará a trabalhar com mais complexidade no desenvolvimento de sistemas de softwares. Como se isso não bastasse, os usuários desses sistemas precisarão de sistemas mais inteligentes, interfaces mais simples e funcionalidades

mais completas. Os construtores de softwares vão precisar lidar com quantidades maiores de detalhes para atender às exigências mais sofisticadas dos usuários. Embora não estatisticamente comprovada, a experiência mostra que, à medida que aumentam o escopo, o tamanho e as dependências dos sistemas de softwares, a complexidade aumenta exponencialmente. Fica evidente que a complexidade é uma preocupação importante que gostaríamos que a arquitetura de softwares conseguisse dar conta.

Há muitas abordagens para a arquitetura de softwares publicadas por autores respeitados na disciplina. Embora eu respeite e admire todos os documentos de pesquisa publicados sobre este tópico, gostaria de concentrar minha lista de pontos principais para tratar da complexidade em apenas dois fatores:

- *A abordagem de cima para baixo no projeto de softwares.* Esse fator enfatiza uma abordagem de cima para baixo na hora de projetar a arquitetura de softwares, começando do sistema que está mais em cima (produto ou grande solução) e dividindo-o em subsistemas que podem ser projetados de forma independente, e depois aproximando cada subsistema iterativamente de uma maneira similar para dividi-lo em componentes independentes.
- *Decomposição ou colapso do sistema.* Esse fator significa projetar os componentes que estão pouco vinculados e têm interfaces claras e intuitivas. A pré-condição para desenvolver componentes adequados é ter uma idéia clara do comportamento dinâmico do sistema e dos subsistemas. Os componentes devem dar conta das exigências funcionais do subsistema abrangente, ajustando-se harmoniosamente com outros componentes.

Essas exigências alimentam a pesquisa e a criação da BSAL (Behavioral Software Architecture Language). Os objetivos principais da BSAL são os seguintes:

- Fornecer um modelo de implementação que possa começar no nível da arquitetura do sistema, permitindo que o arquiteto defina o sistema, os subsistemas, os estados e o comportamento dinâmico do sistema e todos os subsistemas.
- Facilitar uma plataforma integrada para a definição da arquitetura de softwares (estrutura e comportamento) e para a implementação dos softwares (codificação). A BSAL foi projetada para ser a linguagem comum do arquiteto e do desenvolvedor.
- Permitir que o projeto de arquitetura do sistema (inclusive o comportamento) seja reforçado durante a implementação. Pode-se realizar esse reforço com a ajuda dos IDEs (Ambientes Interativos de Desenvolvimento). A separação de funções e os mecanismos de autorização podem ser claramente implementados em um ambiente de desenvolvimento da BSAL.
- Usar uma plataforma que possa ser alterada e aprimorada facilmente por mais contribuições no futuro (a BSAL se baseia no modelo OOD-OOP).

## Por que "comportamental"?

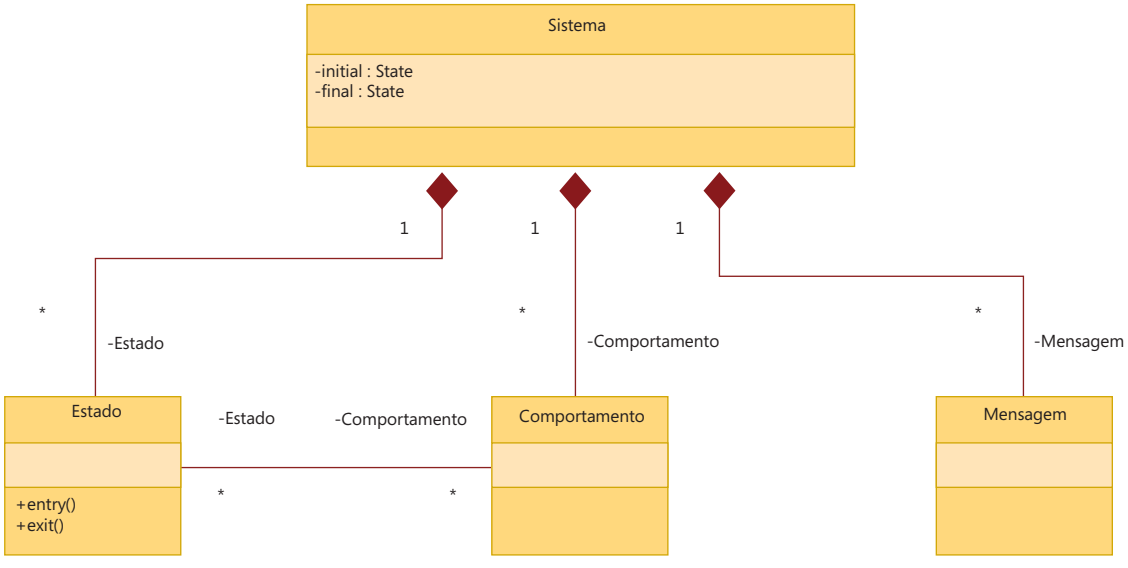
Para entender melhor a importância do comportamento e por que ele foi ampliado na fase de projeto de arquitetura da BSAL, vamos pensar em um cenário do mundo real. Recebemos a tarefa de projetar aprimoramentos importantes no sistema de associação de um site abrangente do tipo B2C (Business-to-consumer). Para tornar a situação ainda mais crítica, suponha que o sistema tenha sido construído por dois ex-funcionários que, atualmente, não estão disponíveis para nos ajudar.

> "UM ARQUITETO EXPERIENTE BUSCARIA COMPREENDER O COMPORTAMENTO DINÂMICO DO SISTEMA À MEDIDA QUE INTERAGE E RESPONDE A OUTROS SISTEMAS E SOLICITAÇÕES DO USUÁRIO, ANTES DE CONSIDERAR AS MUDANÇAS OU OS NOVOS RECURSOS"

Esse sistema normalmente teria recursos para facilitar a criação de novos membros, atualizar as informações dos membros, manipular os direitos desses membros, realizar autenticação e autorização e fornecer "carteiras virtuais" (cartão de crédito, endereço e informações de fatura).

Já que nosso trabalho deve ser responsável por realizar uma revisão técnica no sistema principal de associação, nossa primeira prioridade seria tentar entender esse sistema em sua totalidade. Assim, após examinar a solução de trabalho, buscaríamos documentos e projetos referentes à arquitetura do sistema. Partindo do princípio de que tenhamos sorte o suficiente para encontrar alguns diagramas de classe do sistema, podemos inicialmente nos sentir aliviados.

**Figura 1** O elemento básico dos sistemas



No entanto, após uma investigação e uma análise mais aprofundadas, provavelmente acharíamos que os próprios diagramas são incompletos e desatualizados. Com a frustração, acabaríamos concluindo, na nossa tentativa de entender a arquitetura geral do sistema, que nada, exceto o próprio código, pode revelar a arquitetura desse sistema. Teríamos que arregaçar as mangas e começar a ler o código linha por linha. O código é sempre a fonte definitiva das informações que poderia nos revelar (embora não facilmente) a arquitetura do sistema.

Que informação nos ajudaria mais a visualizar e compreender um sistema de softwares? Embora descobrir as regras de interação e as interfaces do sistema, obter os documentos de análise de negócio desse sistema, ter projetos atualizados do diagrama de classes e obter documentos atualizados das exigências e do projeto técnicos sejam ações que ajudariam, fazer com que o desenvolvedor original do sistema explique o seu comportamento e a seqüência de mudança de estado e condicionamento de eventos do sistema seria a escolha mais favorável.

Essa escolha é favorável porque o comportamento do software e as mudanças de evento/estado são muito difíceis de serem captados sem a ajuda direta dos engenheiros anteriores do sistema. Embora a falta de informações atualizadas seja uma desvantagem definida nesta situação, ser confrontado com um número excessivo de informações também pode levar a um desastre. Ambas as extremidades do espectro de disponibilidade de informações são situações desfavoráveis.

> "A BSAL MESCLA A ARQUITETURA (ESTRUTURA E COMPORTAMENTO) E O CÓDIGO EXECUTÁVEL EM UMA ÚNICA FONTE"

Um arquiteto experiente buscaria compreender o comportamento dinâmico do sistema à medida que interage e responde a outros sistemas e solicitações do usuário, antes mesmo de considerar as mudanças ou os novos recursos. Nessa situação em particular, o comportamento dinâmico do sistema estaria profundamente enraizado na estrutura física da solução (o código-fonte).

É importante salientar que os aspectos comportamentais de um sistema de software desempenham um papel fundamental na hora de revelar a sua natureza real, que é o motivo pelo qual a BSAL inicia a definição de um sistema desse tipo destacando os subsistemas, os estados e os padrões de comportamento.

## Conheça a BSAL

Vamos nos familiarizar com a linguagem simples, porém eficaz, para definir a arquitetura de software (e o código). Embora a BSAL seja uma linguagem de definição de arquitetura, também é uma linguagem de programação. Apesar de haver outros aspectos da arquitetura de software que poderiam ter sido enfatizados, o foco da BSAL encontra-se em fornecer um modelo geralmente simplificado de definição de arquitetura, facilitando um modelo de programação flexível.

Considerando-se que os aspectos comportamentais dos sistemas de softwares sejam os mais importantes, a BSAL enfatiza a definição de padrões de comportamento na fase de definição de arquitetura. Na verdade, sem definir os padrões comportamentais do sistema, você não pode passar para a implementação dos modelos inferiores do sistema. A sintaxe da linguagem de programação BSAL poderia ser qualquer linguagem de programação moderna orientada a objetos.

Embora os princípios da linguagem possam ser aplicados a qualquer linguagem OOP moderna (e certamente espero ver isso no futuro), a C#.NET foi usada aqui para transmitir as idéias da BSAL discutidas. Antes de avançar mais, pode ser útil fornecer um cenário de utilização baseado em funções para a BSAL:

- O arquiteto de software usa a linguagem para definir as principais características do sistema de softwares. Ou seja, o sistema, o subsistema, o estado, o comportamento e os objetos da mensagem são definidos pelo arquiteto. Essas são as definições de componente de alto nível para qualquer solução de software.
- O engenheiro de software usa o resultado direto do trabalho do arquiteto (com o código-fonte de alto nível da BSAL) como os dados de entrada do seu trabalho, fornece um projeto detalhado e começa a implementar os estados, as mensagens, os componentes e as classes dentro dos subsistemas.
- O engenheiro de teste usa o código-fonte da BSAL para analisar o comportamento do sistema e criar cenários de teste em caixas preto-e-branco.
- O analista usa o código-fonte da BSAL para analisar o colapso do sistema de alto nível e as definições de componentes e compreender o comportamento do sistema antes de propor mais aprimoramentos.

A BSAL se baseia no paradigma orientada a objetos. O acréscimo importante que a BSAL põe em jogo é a utilização comum e padronizada de alguns componentes básicos na definição dos softwares. Esses componentes podem, por si só, ser criados com uma linguagem OOP moderna. A regra básica da BSAL é que cada parte do código deve estar dentro de um objeto do sistema. Um objeto de sistema em si pode encapsular sistemas menores ou subsistemas. O objeto de sistema pode conter campos (atributos), deve ter pelo menos dois objetos de estado e pode ter objetos de comportamento e de mensagens. O código executável só pode ser escrito dentro de objetos de estado do sistema, significando que o objeto de sistema (ou o objeto de subsistema de fato), o objeto de comportamento e os objetos de mensagem definem unicamente o comportamento e a estrutura do sistema (a arquitetura). No entanto, quase todo o código executável do programador ficará dentro dos objetos de estado, o que simplifica muito a tarefa do projeto de arquitetura e transporta a arquitetura para dentro do código do programador.

## Colapso do componente

Os componentes de baixo nível da BSAL poderiam ser implementados em qualquer linguagem de programação moderna orientada a objetos, como C#.NET, Java ou C++ (os detalhes desses componentes vão além do escopo deste artigo). A apresentação de um modelo completo de trabalho da BSAL ficará como tema principal de uma discussão futura. Os componentes básicos de alto nível da BSAL são os objetos de sistema, estado, comportamento e mensagem. Esses são os objetos que os arquitetos do sistema geralmente definem. Tais objetos descrevem e definem os limites básicos e as bases do sistema (consulte a Figura 1).

O objeto de sistema é o componente da BSAL que se encontra mais em cima. É o objeto que encapsula todos os outros componentes do sistema. Ele pode ser espalhado em vários arquivos, módulos e/ou pacotes. Um sistema pode ser composto de um ou mais subsistemas. Ele pode encapsular subsistemas, estados, comportamentos, mensagens, campos e atributos personalizados. O sistema deve ter pelo menos dois estados, a saber, o estado inicial e o estado final. Quando o sistema é iniciado, vai imediatamente para o estado inicial; quando se sinaliza para o sistema desligar, vai para o estado final. Os sistemas podem ter um número ilimitado de estados personalizados.

O sistema tem um estado inicial e um estado final; pode receber mensagens (ingresso de dados no sistema); pode verificar as condições comportamentais, para que possa mudar e gerenciar os estados de acordo; pode implementar padrões comportamentais ativando e finalizando estados; e pode enviar mensagens para outros sistemas (saída de dados do sistema):

```
public class member : System
{
protected State
createMember;
public member(
State istate, State
fstate) : base(
istate, fstate)
{
}
}
```

O objeto de estado define um estágio pelo qual o sistema pode passar para realizar uma tarefa específica. As definições de estado organizam as unidades de trabalho de acordo com uma certa ordem lógica. Os estados implementam a funcionalidade principal do sistema.

Um objeto de estado tem dois métodos necessários chamados entry() e exit(). O método entry() é chamado quando o estado é ativado; o método exit() é chamado quando o estado está marcado para conclusão. Um objeto de estado pode ter um número ilimitado de métodos personalizados.

Um sistema pode mudar os estados internamente ou pode mudar o estado em resposta a uma interação externa com outros sistemas ou ambientes. (Por exemplo, receber uma mensagem de outro sistema pode disparar uma mudança de estado em um sistema.) O lugar comum em que é implementada uma mudança de estado dentro de um objeto de comportamento ou outro estado é o método exit().

Um objeto de estado tem um método entry() e um exit(), pode ser ativado por meio de um objeto de comportamento ou outro objeto de estado, pode completar a si mesmo, deve verificar os objetos de comportamento relacionados a ele quando completados, pode ter propriedades e métodos personalizados e pode funcionar como thread separado de execução (estados paralelos):

```
public class MyInitState :
Estado
{
public MyInitState(
string sn, int sid) :
base(sn, sid)
{
}
public override int entry()
{
// faz alguma coisa
return 0;
}
public override int exit()
{
// faz alguma coisa
return 0;
}
}
```

O objeto de comportamento é onde os padrões comportamentais do sistema são definidos. A definição de comportamento encontra-se no centro do gerenciamento de estado de um sistema de software. Em geral, a definição de comportamento contém uma ou mais chamadas de métodos relacionadas a condições para concluir e/ou ativar certos estados do sistema. As definições de comportamento não devem conter nenhum código executável diferente daqueles que resultam em mudança de estado, definem um campo ou atributo ou enviam uma mensagem para outros sistemas. O objetivo único aqui é definir o comportamento do sistema sob certas condições ou circunstâncias.

**"À MEDIDA QUE OS PROBLEMAS DA TECNOLOGIA DA INFORMAÇÃO CONTINUAM A FICAR MAIS COMPLEXOS, OS ENGENHEIROS DE SOFTWARES PRECISAM DE LINGUAGENS (E DE MODELOS DE PROGRAMAÇÃO) QUE POSSAM OCULTAR E ENCAPSULAR NÍVEIS MAIORES DE DETALHAMENTO"**

Um objeto de comportamento basicamente encapsula as condições, as mudanças nos atributos, o envio de mensagens e a conclusão e ativação de estados.

## Bom para a profissão

Um objeto de mensagem encapsula o fluxo de informações síncronas/assíncronas entre sistemas diferentes. Um objeto de sistema só pode receber objetos de mensagem que foram definidos por ele. Uma definição de mensagem em geral contém um ID, um cabeçalho e um corpo de mensagem. A implementação subjacente das mensagens poderia ser deixada para o ambiente e a estrutura específicos em utilização. O padrão comportamental do sistema é verificado quando o sistema recebe qualquer mensagem nova ou (como alternativa) quando qualquer estado dentro da mensagem está sendo concluído.

À medida que os problemas da tecnologia da informação continuam a ficar mais complexos, os engenheiros de softwares precisam de linguagens (e de modelos de programação) que possam ocultar e encapsular níveis maiores de detalhamento. Como discutimos aqui, a analogia do desenvolvimento de OOP tradicional, respaldada pelas anotações da modelagem de softwares, pode apresentar alguns limites e, por vezes, se mostrar enfadonha. A BSAL mescla a arquitetura (estrutura e comportamento) e o código executável em uma única fonte. Os arquitetos e os engenheiros de softwares precisam transmitir, compartilhar e criar idéias. Para propor alternativas ou soluções aprimoradas para os sistemas ativos, eles precisam captar rapidamente a essência dos sistemas de softwares. Ser capaz de entender com rapidez e precisão um sistema de softwares é uma grande virtude. A BSAL é uma resposta possível para essas necessidades e, com otimismo, um passo positivo para abrir novos horizontes no futuro da profissão de engenharia de softwares. •

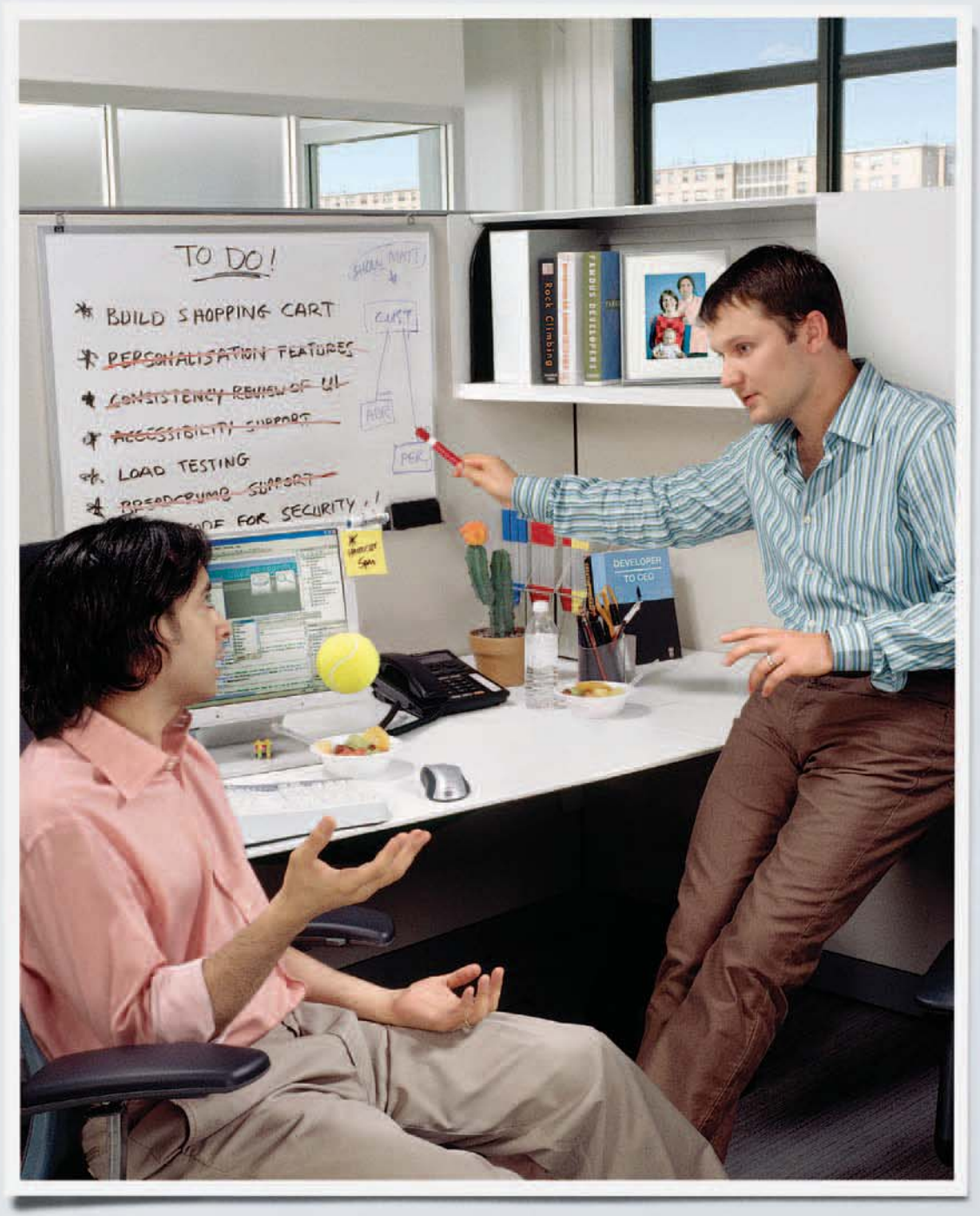
**Sobre o autor**

**Behzad Karim** (MCSD.NET, MCT) é gerente de projeto de softwares no TEPUM SIGMA Consulting and Development Center (www.sigma.net.tr) em Istambul, na Turquia, onde coordena equipes de desenvolvimento de softwares que trabalham em projetos de EAI. Behzad é desenvolvedor e arquiteto de softwares há aproximadamente 18 anos. Entre em contato com Behzad pelo email **bkarim@sigma.net.tr.**

*Your potential. Our passion.™*

**Microsoft®**


**New Visual Studio 2005.**
The difference is obvious.




**Spot the difference?** Once you start coding, you'll see it immediately. The new Visual Studio® 2005 has over 400 new features, such as Web and Windows® controls that reduce tedious tasks and repetition. So you can focus on creating great code. Spot the 10 differences above and play for cool prizes at **msdn.microsoft.com/difference**

Microsoft®
Visual Studio® 2005


Microsoft®

THE ARCHITECTURE JOURNAL™

Idéias para melhores resultados