THE ARCHITECTURE JOURNAL

Journal 5

Intercâmbio de Integração

Microsoft

Sumário

THE ARCHITECTURE JOURNAL
Journal 5

Nota do Editor

Estimado Arquiteto,

Nos últimos 18 meses, tive a felicidade de testemunhar a evolução do *The Architecture Journal* desde sua concepção até esta edição atual: o *Journal 5*. O *Journal* cresceu de inúmeras formas diferentes, em especial como veículo para os arquitetos de todo o mundo compartilharem idéias, aprendizagens e perspectivas exclusivas. Um dos principais indicadores de que este veículo está ganhando vida própria é a idéia de continuidade que emerge dos artigos referências e comentários sobre artigos das edições anteriores e promessas por parte dos autores de escrever novos artigos sobre os tópicos que abordaram nesta edição. Isso me estimulou a explorar as edições anteriores e atiçou minha curiosidade pelas edições futuras.

A arquitetura no mundo dos sistemas é tão ampla quanto uma galáxia, e os arquitetos que fazem uso dela têm desafios únicos como nenhuma outra função que possa me ocorrer agora. O que a torna um desafio ainda maior é que apenas algumas instituições educacionais oferecem um programa formal de graduação nesta disciplina. Ao contrário de suas contrapartes no mundo da construção civil, que têm diplomados emoldurados pendurados nas paredes, o conhecimento de arquitetura não é fácil de adquirir. Trata-se de uma disciplina refinada a qual apenas os bravos Cavaleiros Jedi com aptidão especial escolheram enfrentar.

A rápida taxa de mudança do mundo dos negócios e a velocidade da inovação tecnológica estabelecem que os arquitetos precisam de um meio para se manter atualizados, trocar idéias com os colegas e crescer. Apenas os indivíduos que se sentem confortáveis trabalhando com ambigüidade, conhecimento e experiência em muitas disciplinas, e que gostam de se divertir com as exigências de vários acionistas, tendem a se aventurar nesta fronteira estimulante e exigente. O *Journal* é um meio importante que percorre um longo caminho para tornar isso possível.

Lendo os seis artigos e admirando os gráficos desta edição, aprendi coisas novas e descobri novas perspectivas de encarar questões complicadas que por vezes cheguei a perder o sono durante a noite. O artigo escrito por Richard Veryard e Philip Boxer me fez divagar pelas ruas de uma metrópole ágil que busca paralelos com o espaço de governança da SOA. Passei por uma lista prévia e abrangente de verificação de Anna Liu e Ian Gorton para avaliar as tecnologias para as minhas necessidades de integração baseadas em serviço. Se eu tivesse que escolher as alternativas certas de transporte para os Web Services, pegaria a rota dos "aviões, trens e automóveis" de Simon Guest. Por um momento, achei que estava lendo sobre o planejamento de transporte metropolitano.

Todos os artigos desta edição são ricos e encantadores, estabelecendo o ritmo para o tema do "Pense para Frente, Aprenda Mais, Resolva Agora" do recém-lançado Centro de Recursos do Arquiteto no endereço Microsoft.com *(www.microsoft.com/architecture)*. Tenho certeza de que esta edição fará você pensar, aprender alguma coisa nova e ajudá-lo a se desenvolver, como aconteceu comigo. Espero que também o motive a usar o *Journal* como meio para compartilhar seu conhecimento e suas perspectivas.

Aproveite!

Gurpreet S. Pall

Apresentação

Caro Arquiteto,

Ficou claro nesta segunda edição do *Journal,* que agora se chama *The Architecture Journal*, que a demanda era alta para este tipo de publicação da Microsoft. Como editor, recebi muitos estímulos de amigos, colegas e clientes para transformar essa idéia numa iniciativa duradoura, e não num milagre passageiro. Meu objetivo sempre foi desenvolver uma forte comunidade com as pessoas que pensam da mesma forma em torno deste *Journal.* Para ser aceito no mercado, ele tinha que fornecer uma plataforma reconhecida para os autores expressarem seus pensamentos e visões de uma forma aberta, com o mínimo possível de interferência da Microsoft, mantendo a credibilidade e a viabilidade financeira que o forte suporte da Microsoft oferece.

Sinto-me feliz em dizer que o *Journal* ainda ficará conosco por um bom tempo. Desde a publicação do Journal 4 e a liberação desta última edição, o *Journal 5*, a publicação saiu de sua fase de incubação para se tornar uma parte essencial da estratégia de desenvolvimento da comunidade de arquitetos da Microsoft. O *Journal* apresenta com destaque o Centro de Recursos de Arquitetos da Microsoft (visite o endereço *www.microsoft.com/architecture)*, e agora fornecemos assinaturas gratuitas da versão impressa. Estamos investindo no profissionalismo que o *Journal* exige ao criar uma equipe dedicada à sua produção e publicação, em colaboração com a Fawcette Technical Publications. O objetivo desses avanços é garantir que você, nossos leitores e nossos autores obtenham valor e experiências excepcionais com a leitura do *Journal*.

A edição temporária do *Melhor do Journal* fez uma transição bem-sucedida do formato "grande" original para este formato normal de revista. Milhares de cópias foram distribuídas este ano nas conferências TechEd nos EUA e na Europa! Ao mesmo tempo, estamos vendo muitas oportunidades e um maior interesse em arquitetura, tanto que as palestras sobre arquitetura especializada nessas conferências competem de igual para igual com as palestras tradicionais de desenvolvedores. Trata-se de um momento estimulante neste espaço, e você pode contar com o *Journal* para abrir caminho!

Por fim, você vai perceber que estou escrevendo esta apresentação, e não o editorial, como seria de se esperar.

Bom, é porque o Gurpreet Pall fez um trabalho tão bom com a versão dele da apresentação que decidi trocar.

Como de costume, se tiver interesse em escrever para esta publicação, envie-me uma breve descrição do seu tópico e o seu currículo para *asehmi@microsoft.com*.

Desejando-lhe uma boa leitura!

Arvindra Sehmi

Uma introdução aos Mapas de Tópicos

Kal Ahmed e Graham Moore

Introdução

Este artigo introduz a norma internacional ISO Mapas de Tópicos. O paradigma mapas de tópicos descreve a maneira como relacionamentos complexos entre conceitos abstratos e recursos do mundo real podem ser descritos e permutados usando uma sintaxe XML padrão.

Os mapas de tópicos foram originalmente desenvolvidos no final da década de 1990 como uma maneira de representar estruturas de índices no final do livro, de modo que vários índices de origens diferentes pudessem ser mesclados. Porém, os desenvolvedores rapidamente perceberam que com uma pequena generalização adicional poderiam criar um metamodelo com aplicação potencialmente muito mais ampla. O resultado desse trabalho foi publicado em 1999 como ISO/IEC 13250-Topic Navigation Maps.

Além de descrever o modelo básico dos mapas de tópico e os requisitos para um processador de mapa de tópicos, a primeira edição da ISO 13250 incluía uma sintaxe de intercâmbio baseada em SGML e a linguagem de ligação de hipermídia conhecida como HyTime. A segunda edição, publicada em 2002 [1], adicionou uma sintaxe de intercâmbio baseada em XML e Xlink. Essa é a sintaxe com o suporte mais amplo até o momento nos produtos de processamento de mapas de tópicos e é a sintaxe que iremos descrever neste artigo.

Hoje existe uma série de implementações da norma, tanto em fonte aberta quanto patenteadas, para diversas linguagens e plataformas, incluindo a plataforma .NET.

Conceitos básicos dos mapas de tópicos

A essência dos mapas de tópicos pode ser resumida de forma muito sucinta: Um mapa de tópicos consiste em uma coleção de tópicos, cada um representando algum conceito. Os tópicos estão relacionados uns aos outros por associações, que são combinações de tópicos de tipo n-ary. Um tópico também pode estar relacionado a qualquer número de recursos por suas ocorrências.

A Figura 1 mostra os três conceitos básicos dos mapas de tópicos. Também mostra como a distinção entre os relacionamentos tópico com tópico e tópico com recurso permite uma partição do modelo em um espaço de tópicos que contém somente tópicos e associações entre tópicos e um espaço de recursos que contém os recursos relacionados aos tópicos.

Essa partição é interessante porque permite que um mapa de tópicos desenvolvido para um conjunto de recursos seja adaptado para indexar um conjunto de recursos diferente. Dessa maneira, o mapa de tópicos pode ser considerado como uma forma portátil de conhecimento.

Ao contrário dos modelos específicos de domínio, o modelo de mapa de tópicos não possui um conjunto de tipos predefinido. Em vez disso, autores de mapas de tópicos individuais ou grupos de autores em uma comunidade de prática podem definir o modelo do seu domínio de interesse e compartilhar esses modelos com outros autores de outros domínios.

Figura 1. Tópicos, associações e ocorrências

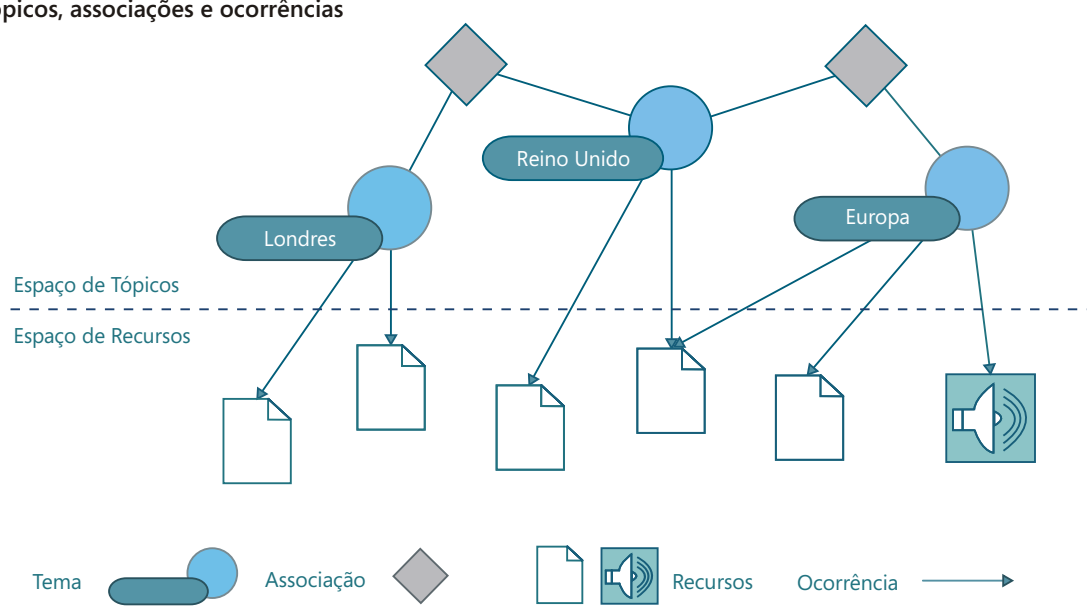
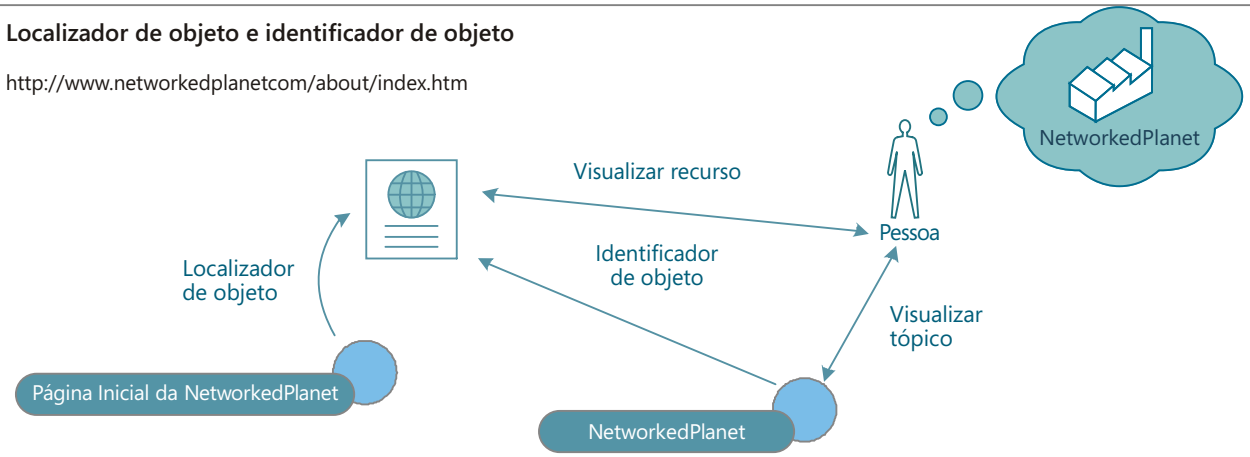


Figura 2. Localizador de objeto e identificador de objeto



Acreditamos que para muitos usuários finais, um bom aplicativo de mapas de tópicos irá ocultar grande parte do mecanismo dos mapas do tópicos, ou todo ele, permitindo aos usuários concentrar-se nos modelos de domínio com que trabalham. No entanto, o modelo de mapas de tópicos e a norma Mapas de Tópicos oferecem uma série de benefícios que podem ser revelados em aplicativos e podem constituir pontos de venda exclusivos.

A metáfora principal dos mapas de tópicos de tópicos, ocorrências e associações alcança um equilíbrio entre ser compacto e fácil de entender e fornecer infra-estrutura básica suficiente para permitir aos usuários converter o seu modelo mental de domínio em um modelo de mapa de tópicos. Outras formas de organização de dados e informações como RDF e o modelo relacional podem possuir um modelo mais simples ainda, mas exigem que o usuário crie infra-estrutura para procedimentos comuns como rotular um item com alguns nomes, definir uma estrutura de classes ou criar relacionamentos n-ary entre itens.

Como descrito acima, o modelo de mapas de tópicos possui uma distinção clara entre o modelo de domínio, expresso como tópicos e associações entre tópicos, e os recursos indexados, expressos como ocorrências que ligam tópicos a recursos. Três benefícios principais podem resultar dessa estrutura:

- O mapa de tópicos pode atuar como uma visão geral de alto nível do conhecimento de domínio contido em um conjunto de recursos. Desse modo, o mapa de tópicos pode servir não somente como um guia para localizar recursos para o especialista, mas também como uma maneira para os especialistas modelarem seu conhecimento de uma forma estruturada. Isso permite aos não especialistas aprender os conceitos básicos e seus relacionamentos antes de aprofundar nos recursos que fornecem mais detalhes.
- Um mapa de tópicos pode ser facilmente segmentado, dependendo dos recursos que estão disponíveis. Alguns editores utilizam um índice baseado em mapa de tópicos de grandes conjuntos de recursos e criam de forma dinâmica o índice apropriado quando publicam um subconjunto desses recursos. Com uma modelagem ponderada é até possível criar diferentes camadas de detalhe em um mapa de tópicos e diferenciar entre produtos de informação baseados na indexação e recursos de navegação que eles fornecem, além do conteúdo informativo dos produtos.
- Mapas de tópicos que indexam diferentes conjuntos de recursos podem ser facilmente combinados. Esse recurso pode ser usado para permitir às organizações "importar" dados e índices de terceiros e integrar sem interrupção seus próprios dados e índices.

Como já observado, a norma Mapas de Tópicos não vem com uma ontologia predefinida. Não existem restrições quanto aos domínios nos quais os tópicos de mapas podem ser aplicados e há relativamente poucas limitações mesmo na abordagem de modelagem tomada.

Intercâmbio Simplificado

Temos visto mapas de tópicos utilizados para modelar relacionamentos temporais entre eventos; relacionamentos entre conceitos abstratos e suas representações; e formas de lógica de primeira ordem, além de relacionamentos mais tradicionais como dicionários de sinônimos, vocabulários controlados e informações comerciais .

Para muitos usuários, o fato de que os mapas de tópicos podem ser intercambiados utilizando uma sintaxe padrão baseada em XML fornece um forte benefício no aprimoramento da portabilidade de seus dados entre aplicativos e plataformas. Além disso, a sintaxe de intercâmbio XML permite uma fácil integração do intercâmbio de informações do mapa de tópicos dentro da arquitetura de Web Services.

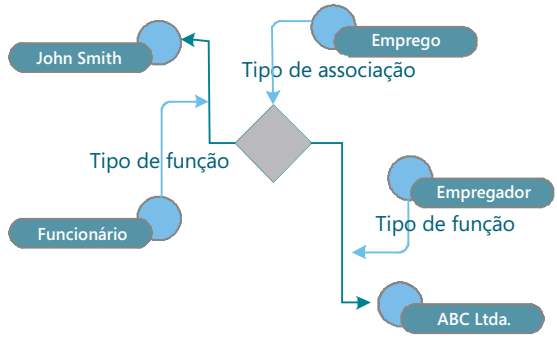
Existem três benefícios principais que os arquitetos e desenvolvedores de sistemas podem obter com a aplicação do paradigma dos mapas de tópicos, que podem ser resumidos como "Flexibilidade, flexibilidade e flexibilidade".

Os mapas de tópicos fornecem o metamodelo no qual um modelo de aplicativo completamente flexível pode ser construído. A criação de novos tipos de objetos comerciais pode ser conseguida adicionando dados à ontologia que constrói o mapa de tópicos.

Como a ontologia é expressa como tópicos e associações entre tópicos, a extensão da ontologia torna-se uma questão de adicionar dados, não uma questão de redesenhar o esquema de armazenamento de base. Isso torna possível modificar o modelo de dados usado por um aplicativo sem a necessidade de atualizar armazenamentos persistentes implantados.

Com os dados do aplicativo armazenados em um metamodelo padronizado e extensível, o caminho está aberto para permitir integração e extensão muito mais simples de aplicativos de terceiros. Esse modelo permitiria a um desenvolvedor de terceiros definir seus próprios dados e extensões de código para um aplicativo sem depender do esquema do aplicativo central que suporta as estruturas de dados específicas da extensão.

Figura 3. A estrutura de uma associação



Além de permitir extensões de terceiros ao esquema do aplicativo, a flexibilidade da estrutura de mapas de tópicos pode ser usada para permitir aos usuários criar suas próprias extensões. Isso tem dois efeitos:

- Permite que os aplicativos sejam altamente personalizáveis: permitindo integração muito mais fácil dos sistemas de dados específicos do cliente, por exemplo. Consideramos isso como sendo a chave para aplicativos verticais que podem ser mais facilmente implantados para vários clientes. Por exemplo, um aplicativo de mapas de tópicos de grande sucesso foi produzido por um editor de informações jurídicas para o mercado de serviços financeiros. O ponto de venda exclusivo desse produto baseado em mapas de tópicos é que é possível integrar a documentação de procedimentos e marketing dos seus clientes com as informações jurídicas que fornecem.
- Isso permite o desenvolvimento de aplicativos horizontais que podem ser integrados mais facilmente nos ambientes existentes.

Um tópico é uma representação de um conceito que pode ser processada por máquina. A norma Mapas de Tópicos não restringe de forma alguma o conjunto de conceitos que podem ser representados como tópicos. Normalmente os tópicos são usados para representar recursos eletrônicos (como documentos, páginas da Web, Web Services) e recursos não eletrônicos (como pessoas ou lugares). Os tópicos podem igualmente ser usados para representar coisas que não possuem forma tangível, como empresas, eventos e conceitos abstratos como "pensões" ou "seguro".

Formas de Identidade

Os tópicos possuem quatro formas de identidade principais. Um tópico pode possuir zero ou mais de cada uma dessas formas de identidade e, desse modo, pode ser identificado dentro de um sistema de mapas de tópicos por um número de diferentes maneiras:

- **Identidade como um recurso de tópico em um mapa de tópicos serializado**. Quando um mapa de tópicos é representado de forma serializada para intercâmbio, cada tópico recebe um identificador URI que é exclusivo nesse mapa de tópicos. Esses URIs são usados principalmente para referências desserializantes entre tópicos. Esses identificadores são denominados localizadores de origem.
- **Identidade como um rótulo que pode ser lido por humano.** Um tópico pode ter qualquer quantidade de nomes de tópico. Os nomes atuam como rótulos para consumo humano e podem ser texto ou uma referência a alguma representação não textual (por exemplo, um ícone, um clip de som, um clip de animação etc.). O mecanismo escopo (descrito mais adiante) permite o caso de homônimo (onde uma única palavra é usada para referir a dois ou mais conceitos diferentes).
- **Identidade por referência.** Quando um tópico for usado para representar um recurso que já possui seu próprio URI exclusivo, esse URI pode ser usado como parte da identidade do tópico. Isso é simplesmente uma maneira de dizer ao agente processador que "Esse tópico representa esse recurso." Na norma de mapas de tópicos, essa forma de identificador é conhecida como localizador de objeto.
- **Identidade por descrição.** Os tópicos podem ser usados para representar um conceito que não possui o seu próprio URI exclusivo. Muitas das coisas que um tópico pode representar nunca poderiam possuir um URI exclusivo porque não são coisas para as quais um computador pode resolver uma referência. Por exemplo, uma pessoa pode possuir qualquer quantidade de registros de banco de dados sobre si mesmo ou biografias ou fotos on-line, mas nenhum desses recursos aos quais se pode dar um endereço é a pessoa, são meramente uma forma de descritor da pessoa. Na norma de mapa de tópicos, essa forma de identificador é conhecida como identificador de objeto e o recurso ao qual o identificador de objeto resolve é conhecido como indicador de objeto. Os mapas de tópicos permitem o uso de referências de URI a esses recursos descritivos como uma forma de identidade. Obviamente é importante que o autor do mapa de tópicos escolha recursos descritivos não ambíguos para essa finalidade; voltaremos mais adiante a essa questão.

A distinção entre as duas últimas formas de identidade pode ser confusa. Considere o URL *http://www.networkedplanet.com/about/index.html*. Essa é uma página da Web que descreve a empresa NetworkedPlanet. Assim, esse URL poderia ser usado como o identificador de objeto de um tópico denominado "A empresa NetworkedPlanet" porque resolve para um recurso que descreve o conceito da empresa. No entanto, se desejássemos falar sobre o conceito "A página 'Sobre' no site *www.networkedplanet.com*," na realidade desejamos um tópico cujo objeto realmente seja o recurso no endereço *http://www.networkedplanet.com/about/index.html*, por isso usaríamos o mesmo URI que o localizador de objeto.

Tipos e Nomes

A diferença principal entre um identificador de objeto e um localizador de objeto é que um identificador de objeto requer interpretação humana como um recurso para determinar o conceito que um tópico representa, enquanto que um localizador de objeto simplesmente aponta para o conceito que o tópico representa. Isso é mostrado na Figura 2. A seta sólida escura mostra o uso de um recurso como localizador de objeto. A seta sólida clara mostra o uso do mesmo recurso como identificador de objeto. As setas pontilhadas finas mostram a função do ser humano na interpretação de um identificador de objeto.

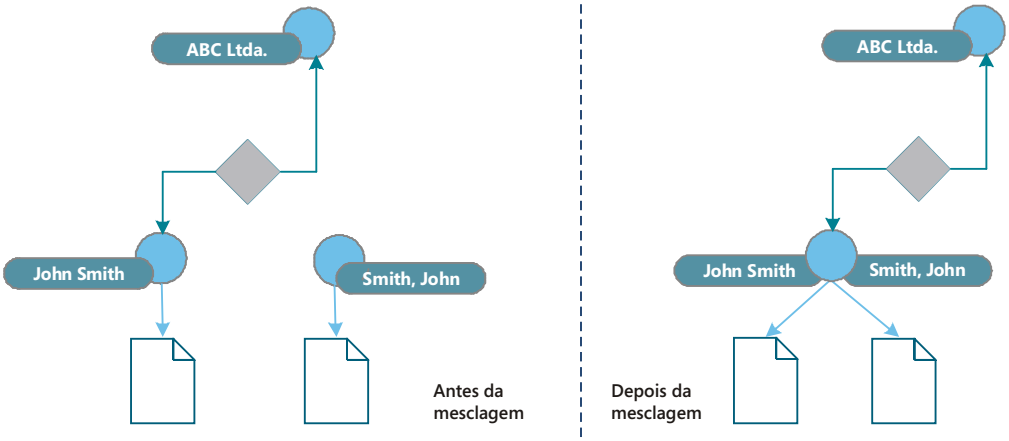
Embora um único tópico possa ter muitas formas de identidade, é importante observar que cada identificador separado pode resolver para apenas um tópico. As regras de mesclagem de mapas de tópicos (descritas mais adiante) aplicam esse relacionamento de um para muitos entre tópicos e seus identificadores.

Além dessas formas de identidade, um tópico também pode possuir qualquer quantidade de tipos e qualquer quantidade de nomes.

Os tipos de um tópico definem a classe (ou classes) de conceito ao qual pertence o conceito representado pelo tópico. Os tipos são tratados nos mapas de tópicos como conceitos; desse modo, todo tipo é representado por um tópico. O tipo de um tópico é especificado simplesmente por uma forma de relacionamento privilegiada entre o tópico que representa a instância e o tópico que representa o tipo.

Os nomes de um tópico definem um conjunto de rótulos de um tópico. Todo nome possui uma estrutura hierárquica. Na raiz está o

Figura 4. Mesclagem de tópicos



nome base, que possui uma representação de seqüência. É o valor de seqüência do nome de base que é usado para determinar a identidade de tópico pelo rótulo. Um nome base também é um contêiner de qualquer quantidade de formas alternativas (conhecidas como nomes variantes). As formas alternativas de um nome podem ser valores de seqüência ou referências a recursos, permitindo que representações como ícones ou clipes de som sejam referenciados como nomes variantes. Os nomes base e nomes variantes podem ter um contexto (ou escopo) no qual são válidos, permitindo que um aplicativo ciente de mapas de tópicos selecione o melhor nome para apresentar a um usuário em uma situação determinada. Esse escopo será abordado mais adiante.

Associações

Associações são a forma geral da representação de relacionamentos entre tópicos em um mapa de tópicos. Uma associação pode ser considerada como um agregado de tópicos n-ary. Ou seja, uma associação é um agrupamento de tópicos sem uma direção ou ordem implícita e não existe restrição quanto ao número de tópicos que podem ser agrupados.

Um tipo (definido por um tópico) pode ser atribuído a uma associação, e especifica a natureza do relacionamento representado pela associação. Além disso, cada tópico que participa da associação desempenha uma função de tipo que especifica a maneira como o tópico participa.

Por exemplo, para descrever o relacionamento entre uma pessoa, "João Silva," e a empresa para a qual ele trabalha, "ABC Limitada," criaríamos uma associação tipificada pelo tópico "Emprego" e com os tipos de funções "Empregado" (para a função desempenhada por "João Silva") e "Empregador" (para a função desempenhada por "ABC Limitada").

Como os nomes, a uma associação pode ser atribuído um escopo no qual seja válida e que pode ser usado por um aplicativo ciente de mapas de tópicos para determinar se deve exibir as informações representadas pela associação a um usuário em uma situação determinada.

Ocorrências

As ocorrências são usadas para representar ou para referir a informações sobre um conceito representado por um tópico. As ocorrências podem ser usadas para armazenar dados de seqüência em um mapa de tópicos ou para referenciar qualquer tipo de recurso endereçável na Web externo ao mapa de tópicos. Nenhuma restrição é colocada sobre qual tipo de recurso é tratado por uma ocorrência. Pode ser uma página HTML estática, uma página HTML gerada por ASP, um Web Service ou qualquer outro tipo de recurso. As ocorrências também não são restringidas ao protocolo HTTP - qualquer endereço codificado como um URI pode ser usado para abordar um recurso externo. Mais uma vez, as ocorrências podem ser tipificadas, usando um tópico para expressar o tipo de ocorrência e um escopo de validade também pode ser atribuído a uma ocorrência.

Escopo

Escopo é o termo usado na norma de mapas de tópicos para referir a uma restrição ou um contexto no qual alguma coisa é dita sobre um tópico. A maneira como essas instruções sobre tópicos são feitas é adicionando um nome ao tópico; especificando uma ocorrência para um tópico; ou criando uma associação entre tópicos (nesse caso a instrução aplica-se a todos os tópicos da associação).

Em muitos casos as instruções não são sempre verdadeiras, mas dependem de um contexto. Por exemplo, fazemos afirmações como "ABC Limitada foi o principal fornecedor de coisas em Q2 2004," ou "Fred diz que ABC Limitada é um bom investimento." Nessas instruções o contexto é mostrado em itálico: um contexto temporal no primeiro caso e um contexto de citação no segundo caso. De forma mais prosaica, o contexto geralmente é usado para facilitar interfaces de vários idiomas, de modo que o conceito "cachorro" possa ter o rótulo "dog" no contexto do idioma inglês, *"le chien"* em francês e *"das hund"* em alemão.

Em um mapa de tópicos, o escopo é definido por uma coleção de tópicos que podem ser atribuídos a um nome, uma ocorrência ou uma associação. O escopo padrão (onde nenhum conjunto é atribuído) é conhecido como o escopo não restringido e significa simplesmente que o nome, a ocorrência ou a associação são sempre válidos.

Quando um aplicativo ciente de mapas de tópicos encontra um nome, ocorrência ou associação que tenha um escopo atribuído, o aplicativo deve utilizar as informações que possui sobre o contexto operacional atual e comparar essas informações com as informações do escopo contidas no mapa de tópicos para determinar se a construção é válida e se deve ser apresentada ao usuário.

Na edição atual da ISO 13250, a mecânica de processamento do escopo em relação a um contexto do aplicativo não é restringida pela norma , sendo que muitos desenvolvedores de mapas de tópicos consideram isso uma desvantagem, pois pode dificultar o intercâmbio de mapas de tópicos que utilizam escopo. A próxima revisão da norma irá recomendar que um escopo que consiste em vários tópicos deve ser processado de forma que a construção do escopo seja válida somente se o aplicativo determinar que todos os tópicos do escopo aplicam-se ao contexto do aplicativo atual.

Lista 1. A sintaxe XML define um elemento topicMap que contém qualquer quantidade de tópicos e elementos de associação. Um exemplo simples de mapa de tópicos em sintaxe XTM é mostrado a seguir.

```
-->
<!-- The Clash is a Band -->
<topic id="clash">
<instanceOf>
  <topicRef xlink:href="#band"/>
</instanceOf>
<baseName>
  <baseNameString>The Clash</baseNameString>
</baseName>
</topic>
<!-- Joe Strummer is a Person (note multiple names)
-->
<topic id="joe-strummer">
<instanceOf>
  <topicRef xlink:href="#person"/>
</instanceOf>
<baseName>
  <scope>
    <topicRef xlink:href="stage-name"/>
  </scope>

  <baseNameString>Joe Strummer</baseNameString>
</baseName>
<baseName>
  <baseNameString>Joseph Mellor</baseNameString>
</baseName>
</topic>
<!-
Joe Strummer is a member of The Clash -->
  Note separate member elements
  used for the different roles
  played
-->

<association>
<instanceOf>
  <topicRef xlink:href="#membership"/>
</instanceOf>
<member>
<roleSpec>
  <topicRef xlink:href="#group"/>
</roleSpec>
<topicRef xlink:href="#clash"/>
</member>
<member>
<roleSpec>
  <topicRef xlink:href="#singer"/>
</roleSpec>
<topicRef xlink:href="#joe-strummer"/>
</member>
<member>
<roleSpec>
  <topicRef xlink:href="#guitarist"/>
</roleSpec>
  <topicRef xlink:href="#joe-strummer"/>
</member>
</association>
</topicMap>
```

Mesclagem de tópicos

Mesclagem automática de tópicos é um recurso importante dos mapas de tópicos que traz muitos benefícios ao desenvolvimento de mapas de tópicos e aos aplicativos que utilizam mapas de tópicos para gerenciar e trocar dados.

O princípio por trás da mesclagem de tópicos é que, em qualquer mapa de tópicos determinado, cada objeto descrito pelo mapa de tópicos deve ser representado por um e apenas um tópico no mapa de tópicos. Isso significa que é responsabilidade do processador de mapas de tópicos tentar identificar a situação em que dois tópicos representam o mesmo objeto e processá-los de forma que somente um tópico permaneça. Esse é o processo de mesclagem.

A identificação de quando dois tópicos representam o mesmo objeto é conseguida aplicando-se heurística. A norma de mapas de tópicos define um conjunto de heurística básica:

1. Se dois tópicos compartilharem o mesmo localizador de origem, eles foram analisados da mesma origem de mapa de tópicos e devem ser considerados como representando o mesmo conceito.
2. Se dois tópicos possuírem o mesmo localizador de objeto, ambos identificam o mesmo recurso de rede como sendo o que eles representam.
3. Se dois tópicos possuirem o mesmo indicador de objeto, ambos estão utilizando o mesmo recurso para descrever o conceito que representam e devem ser considerados como representando o mesmo conceito.
4. Se dois tópicos possuirem um nome base cada um com a mesma representação de seqüência e o escopo dos nomes base for o mesmo conjunto de tópicos, os tópicos devem ser considerados como representando o mesmo conceito.
5. Finalmente, um aplicativo de mapa de tópicos pode utilizar qualquer informação específica de domínio que possua para determinar se dois tópicos representam o mesmo conceito.

O item (3) na lista acima ressalta a importância de selecionar um bom recurso como a descrição de um conceito. Se a descrição for de algum modo ambígua ou se o recurso tratado não estiver suficientemente definido, é possível que dois autores de mapas de tópicos diferentes possam usar o mesmo recurso como um descritor de conceitos diferentes, resultando em mesclagem indesejada. Na nossa experiência, os bons recursos para descritores de objeto são os criados especificamente para descrever um único objeto: as páginas em wikipedia.org, por exemplo, ou páginas criadas pelos autores de mapas de tópicos ou por uma comunidade de praticantes para definir um vocabulário controlado.

O item (4) mostrou-se controverso na comunidade dos mapas tópicos, pois se apóia no que muitos consideram uma forma de identidade relativamente fraca: o nome de um conceito em algum idioma. O mapeamento de palavras de um idioma para conceitos é um assunto complexo, pois existe o desafio de várias palavras possuindo significados diferentes (homônimos), para não mencionar os desafios de localização! Na próxima versão da norma ISO, as restrições à identidade baseada em nome serão mais rígidas ainda para exigir que um autor sinalize explicitamente um nome de tópicos como sendo o que deve ser usado para conferir uma identidade (o padrão sendo que um nome não irá conferir identidade ao seu tópico).

O item (5) permite que os aplicativos estendam o conjunto de critérios de mesclagem da norma Mapas de Tópicos com critérios específicos do aplicativo. Poderiam incluir critérios baseados em mais de uma seqüência direta ou comparação de URIs. Por exemplo, um aplicativo poderia saber que "O Duque" e "John Wayne" são nomes do mesmo ator e mesclar dois tópicos nessa base. Tendo identificado os tópicos a ser mesclados, o processo de mesclagem define o processo de substituir esses dois (ou mais) tópicos por um único tópico. O tópico único que resulta do processo de mesclagem possui todos os identificadores, nomes (incluindo nomes variantes) e ocorrência dos tópicos que são mesclados. Além disso, o tópico resultante substitui os tópicos mesclados toda vez que eles forem referenciados (ou seja, em quaisquer associações, escopos ou tipos em que apareçam). Esse processo é mostrado de forma esquemática na Figura 4.

A mesclagem com dois ou mais mapas de tópicos é simplesmente o processo de combinar seus conjuntos de tópicos e associações e, em seguida, aplicar as regras de mesclagem de tópicos ao resultado.

Intercâmbio e a sintaxe XTM

Como citado acima, a norma ISO Mapas de Tópicos define duas sintaxes de intercâmbio padrão, uma baseada em SGML e a outra baseada em XML. A sintaxe XML define um elemento topicMap que contém qualquer quantidade de tópicos e elementos de associação. Um exemplo simples de mapa de tópicos em sintaxe XTM é mostrado a seguir:

```
<topicMap xmlns=
  http://www.topicmaps.org/
  xtm/1.0/ xmlns:xlink=
  "http://www.w3.org/1999/
  xlink">
  <topic id="band">
  <baseName>
   <baseNameString>Band
   </baseNameString>
  </baseName>
  </topic>
  <topic id="person">
  <baseName>
   <baseNameString>Person
   </baseNameString>
  </baseName>
  </topic>
<!--
```

Não entraremos nos detalhes da sintaxe aqui. O leitor interessado é direcionado à especificação original Mapas de Tópicos XML [2] produzida por TopicMaps.org (que foi depois adotada pela ISO).

Deve ser observado que a sintaxe XTM não impõe as restrições de mesclagem que são requeridas de um processador de mapas de tópicos. Isso permite que XTM seja criado facilmente, mas requer que qualquer processador capaz de ler um arquivo XTM deva detectar tópicos que devem ser mesclados e aplicar as regras de mesclagem durante a análise do arquivo XTM. Quando um arquivo XTM for considerado "totalmente mesclado" (ou seja, não contém elementos de tópicos representando tópicos que devem ser mesclados), o modelo de mapa de tópicos que ele contém pode ser facilmente acessado com o uso de ferramentas de processamento XML padrão como XSLT and Xquery. No entanto, não é o caso de que ferramentas de processamento XML padrão possam ser facilmente aplicadas a arquivos XTM onde for necessário mesclagem.

Apesar dos problemas com mesclagem, a sintaxe XTM serve a necessidade básica de permitir intercâmbio entre aplicativos de processamento de mapas de tópicos conformativos. Além disso, a sintaxe e as regras de mesclagem juntas são flexíveis o suficiente para permitir que partes de um mapa de tópicos sejam serializadas como documentos XTM separados e posteriormente recombinadas por meio de mesclagem [3].

Padrões de mapas de tópicos

Como esperamos ter demonstrado até este ponto, a norma Mapas de Tópicos fornece uma arquitetura de base bastante flexível para uma ampla variedade de aplicativos de gerenciamento de conhecimento e informações. Essa flexibilidade pode levar a confusão e à reinvenção constante de abordagens básicas de modelagem. Para tratar dessa questão, defendemos o desenvolvimento e uso de padrões dentro de aplicativos de mapas de tópicos. Dividimos os padrões em duas categorias amplas: Padrões de Design de Mapas de Tópicos que são padrões de modelagem de dados de mapas de tópicos; e Padrões de Aplicativos de Mapas de Tópicos que são padrões arquitetônicos para o uso de sistemas de processamento de mapas de tópicos.

O conceito básico de um Padrão de Design de Mapa de Tópicos apóia-se bastante nos padrões de design da engenharia de software. Um Padrão de Design de Mapa de Tópicos fornece uma ontologia focalizada e reutilizável que trata de um único problema. Mas existe um par de diferenças interessantes.

- Um Padrão de Design de Mapa de Tópicos pode ser determinado com mais precisão do que um padrão de design de software, pois deve especificar os URIs dos identificadores de objeto dos tópicos principais usados pelo padrão. Dessa maneira, toda implementação de um padrão específico em um mapa de tópicos pode ser reconhecida instantaneamente pela presença de tópicos com os URIs especificados pelo padrão.
- Como um mapa de tópicos é puramente constituído por dados, comportamentos relacionados a um Padrão de Design de Mapa de Tópicos são implementados não no próprio mapa de tópicos, mas no software de processamento que faz uso dos dados do mapa de tópicos. Alguns padrões de design podem aconselhar um conjunto de comportamentos específico para aplicativos de processamento; outros podem descrever apenas o modelo de dados e deixar em aberto a maneira como o aplicativo processa o modelo de dados.

Alguns padrões básicos retirados da Library Science foram definidos por um dos autores e recebem suporte de uma série de aplicativos de processamento de mapas de tópicos [4]. Esses padrões incluem a classificação hierárquica e de facetas. Mostramos aqui um exemplo desse padrão.

O Padrão de Classificação Hierárquica utiliza uma propriedade de modelagem bastante útil dos mapas de tópicos. Ou seja, que todo tipo de tópico, associação e ocorrência é ele próprio um tópico. Esse recurso permite que a ontologia de um aplicativo de mapa de tópicos seja anotada com o uso da mesma estrutura que é usada para preencher a própria ontologia e pode ser usado para grandes efeitos nos padrões de design, permitindo que uma ontologia de mapa de tópicos existente seja anotada usando o padrão "metaontologia" em vez de modificado.

Esse padrão permite a um aplicativo processar um conjunto de associações entre tópicos como representando uma hierarquia.

Padrões de Aplicativos de Mapas de Tópicos

Os Padrões de Aplicativos de Mapas de Tópicos fornecem padrões arquitetônicos de alto nível e, principalmente, concentram-se na integração de um sistema de processamento de mapas de tópicos com outros sistemas de dados e aplicativos. Esses padrões incluem padrões para representar informações de sistemas de dados externos como dados de mapas de tópicos; padrões para a importação de informações de sistemas de dados externos; e padrões para a exportação e exibição de dados de mapas de tópicos.

Discutiremos os aplicativos de mapas de tópicos com mais detalhes em um próximo artigo.

No momento em que escrevemos, mais trabalho está sendo feito na ISO, tanto na norma Mapas de Tópicos quanto em um conjunto de normas acompanhantes

Embora a ISO/IEC 13250 tenha passado por uma revisão, o núcleo da norma permanece inalterado desde 1999, um grau de estabilidade razoável em comparação com muitas normas de Internet. No entanto, o comitê da ISO decidiu que a próxima versão da norma será uma retificação significativa na maneira como a norma é apresentada e uma retificação secundária da norma em si.

A norma ISO/IEC 13250 deverá ser dividida em uma série de partes distintas: Uma introdução não normativa; uma descrição formal do modelo de dados de base dos mapas de tópicos; uma sintaxe de intercâmbio baseada em XML/Xlink com uma descrição do processo de desserialização da sintaxe em uma instância do modelo de dados e serialização do modelo de dados em um documento que conforma à sintaxe de intercâmbio; e um algoritmo de concessão para o modelo de dados que pode ser usado em testes de conformidade do processador de mapas de tópicos. Espera-se que essa organização torne a norma mais acessível ao leitor e acrescente recursos que faltavam originalmente e foram percebidos como importantes para desenvolvimentos futuros (em particular, a especificação de modelo formal e o algoritmo de concessão).

As alterações na norma incluem a capacidade de aplicar tipos de dados a valores de ocorrência, incluindo a capacidade de inserir XML; a capacidade de declarar um subconjunto dos nomes de um tópico como nomes a serem usados para determinar a identidade do tópico; um modelo de escopo mais claro; e uma definição da sintaxe de intercâmbio em W3C XML Schema e Relax-NG, além de XML DTD

Além das alterações na ISO/IEC 13250, o comitê também iniciou trabalhos em duas normas acompanhantes. ISO/IEC 18408: TMQL (Topic Maps Query Language) definirá uma linguagem para consultar o modelo de dados de mapas de tópicos, permitindo a seleção de construções de mapas de tópicos (como tópicos e associações) e dos dados transportados por eles (nome do tópico ou valores de ocorrência, por exemplo).

Padrão de Classificação Hierárquica

Demonstrativo de problema

Muitos sistemas de classificação consistem em uma ou mais hierarquias dos objetos. Várias diferentes relações hierárquicas são possíveis entre parte e todo, mais amplo e mais restrito, e assim por diante. Embora as relações possam ser diferentes, a semântica hierárquica permanece a mesma. Um aplicativo que esteja lidando com múltiplos tipos de relacionamento hierárquico precisa de uma forma para identificar todos esses tipos.

Descrição do padrão

O padrão apresentado aqui para modelar um sistema de classificação hierárquica usa um tópico para representar cada classe no sistema. A hierarquia, então, é modelada criando-se associações entre classes subordinadas e superordinadas. Entretanto, reconhece-se que há uma ampla variedade de diferentes relações hierárquicas. Por esse motivo, o tipo de associações entre as classes subordinadas e superordinadas não são definidas por esse padrão. Em vez disso, esse padrão define o tipo de todos esses tipos, e o tipo de todos os tipos de funções para as funções subordinadas e superordinadas.

A outra questão é como relacionar os itens classificados de acordo com esse esquema (as instâncias) e os tópicos que representam as classes. Se uma instância for representada por um tópico, então deve ser feita uma associação entre o tópico representando a classe e o tópico representando a instância. Para esse propósito, são introduzidos os tipos de tópico para representar a classificação de uma instância ("Classificado Como") e as funções desempenhadas nessa relação ("Classificação" e "Instância"). Se a instância não estiver representada como um tópico, então deve ser usada uma ocorrência, e nesse caso o tipo de "instância" pode ser usado como um tipo de ocorrência mais que como um tipo de função de associação.

Análise

O padrão cria um meio de sinalizar um tipo de associação como sendo uma relação hierárquica, e de indicar qual função é a superordinada e qual é a subordinada. Isso pode ser feito externamente para o mapa de tópicos que define os tipos de associação e de função, permitindo que um mapa de tópicos preexistente seja integrado sem a necessidade de editá-lo.

A semântica de classificação dos tipos "Classificado Como", "Classificação" e "Instância" podem ser omitidos desse padrão, no qual a classificação não é o propósito da hierarquia. Por esse motivo, esses objetos estão definidos em um conjunto separado de PSI (com uma base URI diferente) dos objetos definidores da hierarquia.

PSIs para o padrão de classificação hierárquica

Os Identificadores de Objetos Publicados a seguir são usados pelo Padrão de Classificação Hierárquica.

Tipo de relação hierárquica

http://www.techquila.com/psi/hierarchy/#hierarchical-relation-type

Um tipo de tipo de associação. As associações que são classificadas em tipos por um tópico que é uma instância desse tipo representam uma relação pai-filho entre dois ou mais tópicos.

Tipo de função superordenada

http://www.techquila.com/psi/hierarchy/#superordinate-role-type

Um tipo de tipo de função de associação. O(s) participante(s) de uma função que é classificada em tipos por uma instância desse tipo em uma associação do tipo Tipo de Relação Hierárquica é(são) o elemento superior na hierarquia.

Tipo de função subordinada

http://www.techquila.com/psi/hierarchy/#superordinate-role-type

Um tipo de função de associação. O(s) participante(s) de uma função que é classificada em tipos por uma instância desse tipo em uma associação do tipo Tipo de Relação Hierárquica é(são) o elemento subordinado na hierarquia.

Classificado como

http://www.techquila.com/psi/classification/#classifed-as

Um tipo de associação que afirma as relações entre um tópico que representa uma classe em um sistema de classificação (desempenhando a função Classificação) e um ou mais tópicos que representam instâncias dessa classe (desempenhando a função Instância).

Classificação

http://www.techquila.com/psi/classification/#classification

A função desempenhada por um tópico que representa uma classe em um esquema de classificação.

Instância

http://www.techquila.com/psi/classification/#instance

A função desempenhada por um tópico que representa um objeto que é classificado conforme um esquema de classificação.

ISO/IEC 19756: TMCL (Topic Maps Constraint Language) define uma linguagem de esquema para mapas de tópicos que permitiria ao autor do esquema restringir as construções que podem aparecer em um mapa de tópicos e a maneira como elas devem relacionar-se umas com as outras. Do mesmo modo que XML, uma linguagem de esquema para mapas de tópicos permite validação e também aplicativos de edição mais inteligentes dirigidos por esquema.

Essas duas normas estão atualmente em um estágio inicial de desenvolvimento com requisitos definidos e, no caso da TMQL, foi criada uma proposta inicial para a linguagem.

O trabalho sobre a norma central e as linguagens de consulta e restrição pode ser seguido no site ISO Topic Maps [5].

Resumo

Este artigo introduziu o paradigma dos mapas de tópicos no contexto da norma ISO. Apresentamos os componentes principais do modelo de mapas de tópicos, mostrando como os componentes de processamento padrão de escopo e mesclagem de tópicos fornecem poder adicional a esse modelo.

Em um próximo artigo apresentaremos alguns casos de uso concretos de mapas de tópicos e mostraremos como o modelo de mapas de tópicos pode ser usado para tratar muitas das necessidades de organização e intercâmbio de informações de um ambiente comercial moderno.

Notas de rodapé

1. A palavra ontologia neste contexto significa o sistema de tipos de tópicos, ocorrências e associações que juntos definem as classes e de coisas e os relacionamentos entre coisas que são documentadas por um mapa de tópicos.

Referências

1. Biezunski M., Newcomb S., Pepper S. (ed.). ISO/IEC 13250:2002, Topic Maps [on-line]. ISO. Formato PDF. *http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf*

2. Moore G., Pepper S. (ed.), XML Topic Maps (XTM) 1.0 [online], TopicMaps.Org. Formato HTML. *http://www.topicmaps.org/xtm/1.0/*

3. Ahmed K., TMShareTopic Map Fragment Exchange in a Peer-To-Peer Application. Formato HTML. *http://www.techquila.com/topicmapster.html*

4. Ahmed K., Topic Map Design Patterns for Information Architecture. Formato HTML. http://techquila.com/tmsinia.html

5. *http://www.isotopicmaps.org/*

Sobre os autores

Kal Ahmed trabalhou em gerenciamento de informações SGML e XML durante dez anos, fazendo desenvolvimento de software e consultoria . Ele é bem o conhecido na comunidade de mapas de tópicos pelo seu trabalho sobre o kit de ferramentas de mapas de tópicos Java de fonte aberta, TM4J*(http://www.tm4j.org/)* e pelas suas contribuições no desenvolvimento da norma ISO. Kal publicou muitos artigos sobre mapas de tópicos e temas relacionados a mapas de tópicos e faz palestras com freqüência.
Kal é co-fundador da Networked Planet Limited, uma empresa que desenvolve ferramentas de mapas de tópicos e aplicativos baseados em mapas de tópicos para a plataforma .NET.

Graham Moore trabalha há oito anos nas áreas de gerenciamento de informações, conteúdo e conhecimento como desenvolvedor, pesquisador e consultor . Ocupou cargos de proeminência como Diretor de Tecnologia da STEP, Vice-presidente de Pesquisa e Desenvolvimento da empolis GmbH e Cientista-chefe da Ontopia AS. Foi responsável pelo desenvolvimento de produtos de gerenciamento de conhecimento, incluindo K42 Topic Map Engine, X2X Link Management Engine e e:kms Knowledge Suite. Graham é co-editor da norma XTM 1.0 XML Topic Maps e ISO13250-1 e -2 (Topic Map Data Model and Syntax), também é co-editor da TMCL (Topic Map Constraint Language).
Graham é atualmente co-fundador da Networked Planet Limited.

Metropolis e a Governança da SOA

Richard Veryard e Philip Boxer

Introdução

Do tempo ao espaço, a tecnologia definidora dos últimos 1000 anos foi o relógio e o imperativo tecnológico prevalecente tem sido economizar tempo, fazer as coisas irem mais rápido ou melhor do que antes.

Os relógios mecânicos modernos com pesos que caem foram inventados cerca de 1000 anos atrás. A invenção geralmente é atribuída ao papa Silvestre II, que foi um renomado matemático e cientista antes de tornar-se papa. Sob a Regra de São Benedito, os monastérios medievais usavam o relógio para controlar o trabalho e as orações. Lewis Mumford relaciona as origens da Revolução Industrial à Regra de São Benedito e ao domínio do relógio. O filme Tempos Modernos de Charles Chaplin mostra (de forma exagerada) o poder cruel do relógio sobre o operário na linha de produção. A engenharia dos processos de negócios no final do século XX concentrava-se em reduzir o ciclo de trabalho, eliminando a espera. O slogan principal: Just-in-time (na hora certa).

Naturalmente, é muito cedo para dizer qual será a tecnologia definidora dos próximos 1000 anos. Mas já existem alguns sinais de uma mudança da ênfase no tempo para uma ênfase na diferença, A Internet, por exemplo, cria novos tipos de diferenças nos relacionamentos entre pessoas e organizações; as organizações comerciais operam como nós ou conjuntos diferenciados dentro de complexos ecossistemas interligados; a coesão social e institucional é mapeada em relação às coordenadas de complexas dimensões abstratas de diferença.

O nosso próprio entendimento de complexidade baseia-se no reconhecimento de que após transpormos um determinado limite de diferença nos comportamentos dos sistemas, fica impossível prever seu comportamento composto. Por isso, na economia de serviço, esperamos que surjam sistemas orientados a serviço cada vez maiores e mais complexos, mas que sejam também capazes de comportamentos cada vez mais diferenciados. Como iremos ver, esse é um dos principais desafios da SOA (Arquitetura Orientada a Serviços).

A cidade ou metrópole é outro grande sistema complexo, que muitos de nós encontramos em nossas vidas cotidianas e que normalmente experimentamos como diferenciado em seu comportamento. A nossa familiaridade com as cidades torna a metrópole um bom lugar para começarmos a desenvolver a melhor abordagem para construir e controlar esses grandes sistemas complexos. Além disso, muitos dos problemas principais do design e controle de sistemas grandes e complexos orientados a serviços também surgem no campo do design urbano, onde têm sido debatidos há décadas (embora sem obter um consenso).

Por acaso, a contribuição de peso mais recente para o debate sobre arquitetura física e design urbano vem de Christopher Alexander, cujo trabalho há muito aguardado sobre a Natureza da Ordem foi finalmente publicado. Alexander exerce uma profunda influência na engenharia de software há muitos anos; o seu trabalho precursor sobre a Síntese da Forma influenciou os métodos estruturados de Yourdon e deMarco, enquanto que o seu trabalho intermediário sobre Padrões foi acolhido com entusiasmo pelos engenheiros de software, principalmente no mundo OO.

Das cidades à SOA

No século XX, dois dos melhores escritores sobre a natureza das cidades foram Lewis Mumford e Jane Jacobs. Mumford pensava que uma cidade bem ordenada precisava de infra-estrutura e planejamento central, enquanto que Jacobs tomava uma posição mais anarquista.

Aqui estão alguns dos problemas levantados no debate sobre cidade [1]:

- **Adaptabilidade:** Na Inglaterra do século XIX, Manchester estava altamente adaptada à indústria do algodão, mas não conseguiu adaptar-se às ondas de industrialização mais recentes. Por outro lado, Birmingham era muito mais adaptável e isso lhe permitiu favorecer uma série de inovações industriais.
- **Complexidade:** Uma cidade contém uma imensa quantidade de interação social e comercial. Uma cidade vigorosa permite a ocorrência de muitos níveis diferentes de interações e o surgimento de conjuntos e subconjuntos significativos que

Tabela 1. Questões de governança da SOA de Pat Helland

(Em contraste com a governança de cidades metropolitanas...) A governança de TI não é tão madura.	As empresas talvez aprendam muito olhando a forma como as cidades administram o difícil processo da alocação de recursos.
Quem faz as escolhas difíceis em TI? É o CEO, o CIO, os chefes das unidades de negócios, os técnicos ou talvez os comitês?	A projeção mostra quais propostas devem se pagar?
As prioridades são estabelecidas com base no custo, na flexibilidade ou na utilização de ativos?	Qual é o cronograma e a análise de risco em torno dessas projeções?
O que é o sucesso e como é medido?	O que é sagrado na sua organização?
Estamos buscando reduções de custo, transparência no processo comercial ou vantagem competitiva?	Que recursos permanecem depois do financiamento desses esforços?
	Que equilíbrio de investimentos especulativos, de curto prazo e de longo prazo é o correto na cultura corporativa específica?
	Esses problemas são comuns para ambientes metropolitanos e de TI.

- **Controle:** O planejamento da cidade requer a orquestração de desenvolvimentos grandes e pequenos, equilibrando a iniciativa e autonomia local com a coerência global.

Existem alguns fortes paralelos entre planejamento de cidade e arquitetura orientada a serviços, o que torna conveniente transpor idéias e experiências do design urbano para a área da SOA.

- **A distribuição do design:** Decisões de design detalhadas são tomadas por diferentes organizações, cada uma seguindo a sua própria agenda (metas comerciais ou políticas, por exemplo).
- **A constância da mudança:** Os elementos do todo estão sendo constantemente redesenhados e reconfigurados e novos elementos são constantemente adicionados. As estruturas devem se desenvolver de maneiras robustas.
- **A necessidade de aprimoramento progressivo:** Cada incremento de design deve fazer não apenas aprimoramentos locais, mas deve exercer um efeito positivo no todo.
- **A natureza recorrente da arquitetura:** Tarefas de design semelhantes devem ser realizadas em diferentes escalas (níveis de granularidade).

Helland sobre a metrópole

Em um artigo recente em Microsoft Architects Journal *(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmaj/html/aj2metrop.asp)*, Pat Helland oferece uma extensa análise comparando o planejamento e gerenciamento de TI com o planejamento e gerenciamento de cidades. Ele argumenta que o controle da TI tem muito a aprender com o controle da cidade (Tabela 1) e levanta alguns paralelos interessantes entre design urbano e SOA (Tabela 2).

O artigo de Helland levanta os argumentos a seguir:

- O progresso requer padronização. (Segundo Helland, as pessoas nem mesmo lavavam corretamente até terem roupas padrão.)
- A padronização está associada à transformação dos elementos em mercadoria.
- A padronização requer concentração de poder (e se isso desenvolver distorções patológicas de relacionamentos sócio-econômicos, que seja).
- A infra-estrutura requer investimento central. (Uma vez que podemos considerar a infra-estrutura como um ato de padronização local, segue-se que ela deve envolver concentração de poder.)
- O investimento central preserva o "sagrado".

Vamos olhar em detalhe as etapas dos argumentos de Helland.

Padronização. O progresso deve incluir o enriquecimento das vidas das pessoas, enquanto que muitos atos de padronização, pelo contrário, as empobrece. Nem todos estão dispostos a considerar a Wal-Mart como a epítome do progresso. Os sistemas de vida são heterogêneos.

A Utopia de Helland parece ser relativamente homogênea. Os cidadãos até mesmo cheiram igual. A exclusão de odores anti-sociais é conseguida por meio de roupas padrão que podem ser trocadas (embora, sem dúvida, sabão e água limpa também contribuam).

Os métodos de produção modernos propiciam a padronização em massa. Isso envolve a separação da produção em duas camadas: uma camada homogênea de produção em massa e uma camada heterogênea de personalização.

A Natureza da Ordem: Manifesto de Christopher Alexander

A Natureza da Ordem (consulte os recursos no endereço *www.ftponline. com*) — em seus quatro volumes completos — tem implicações para o envolvimento das pessoas no projeto de edifícios e na forma detalhada na qual esse envolvimento provavelmente será bem ou malsucedido. Ela tem implicações para o fluxo de dinheiro. Tem implicações para o manuseio dos detalhes arquiteturais e para a integração bem-sucedida da engenharia estrutural no sistema do projeto. (...) Ela afeta praticamente todas as partes da profissão que hoje conhecemos como arquitetura, e indica a necessidade da mudança, em quase todas elas. Não há dúvida de que, sob o impacto desta teoria, a arquitetura será profundamente alterada, e mudará para melhor.

No design urbano, o elemento padronizado é o que fica debaixo do pavimento: utilidades padrão como água e energia, por exemplo. O elemento humano fica acima do pavimento. É uma questão de controle de a cidade decidir o que deve/pode ficar debaixo do pavimento.

A articulação de um sistema complexo em duas camadas (uma homogênea, uma heterogênea) é um problema arquitetônico. Obviamente, existem empresas com interesses comerciais significantes nessa questão. Por isso, vale a pena suspeitar de uma articulação apresentada como resultado de alguma tradição histórica ou um imperativo técnico [3]. Julgamentos arquitetônicos de aparência puramente tecnocrática geralmente escondem uma pauta comercial. Uma das funções do controle é manter um "campo de atuação equilibrado" entre diferentes pautas comerciais. Por essa razão, os controladores geralmente procuram intervir no nível arquitetônico.

É concebível que algum elemento possa parecer homogêneo do lado do suprimento e heterogêneo do lado da demanda ou vice-versa, de modo que o limite é relativo a uma formulação de suprimento /demanda. A tecnologia está constantemente criando divisões homogêneas e heterogêneas, por exemplo, a tecnologia Voz por IP cria uma divisão entre dispositivos que ficam atentos se um fluxo de bits representa voz ou dados (e, portanto, consideram o tráfego como heterogêneo) e dispositivos que não se importam (e, portanto, consideram o tráfego como homogêneo).

Uma base plausível de articulação das camadas é o índice diferencial de mudança. Pode parecer que faz sentido padronizar e controlar a camada de movimento lento e permitir maior diversidade na camada de movimento rápido. Mas uma perspectiva ecológica nos informa o que a de movimento lento domina a de movimento rápido. Isso implica em uma nova função para a arquitetura:

- Manter a estratificação apropriada das camadas e a ligação entre os elementos dentro e através das camadas e
- Operar em um nível de abstração superior, implementando diretivas de design baseadas em evidências.

As abordagens existentes de definição das arquiteturas podem não funcionar muito bem, mesmo para os bits homogêneos. Elas certamente não funcionam para os bits heterogêneos e também não ajudam a definir os limites entre as camadas homogêneas e heterogêneas. Uma conseqüência do nosso argumento é que os limites entre homogêneo e heterogêneo como ilustrado acima é um foco de atenção apropriado da arquitetura. Voltaremos a falar sobre o que isso significa na prática.

Transformação em mercadoria. Não podemos evitar a transformação das nossas vidas em mercadoria, mas devemos estar cientes dos perigos [4].

Felizmente, não precisamos mais suportar software "de tamanho único". O software situado (software criado para uma situação ou contexto social específico) resiste à pressão tradicional da engenharia de software em direção à generalização e aparentemente desconsidera a economia de escala/escopo. Em vez disso, trabalha unicamente dentro de um sistema sócio-técnico colaborativo (a "comunidade"); as condições para o sucesso do software (incluindo significado e confiança) são criadas conjuntamente pelos membros da comunidade.

Uma das primeiras formas de software situado foi a planilha eletrônica. Usuários avançados construíam estruturas complicadas utilizando Visicalc, Lotus 123 ou Excel. Esses eram essencialmente artefatos não transferíveis com muitas suposições ocultas, mas serviam uma finalidade útil dentro de um contexto determinado. Isso ilustra o fato de que software situado é auxiliado pela existência de ferramentas e plataformas que fornecem suporte generalizado para software situado.

O sucesso devastador da planilha eletrônica deveu-se ao fato de que ela executava uma função útil enquanto deixava o usuário livre para criar significados específicos para o contexto. Mas a planilha eletrônica também era limitada pelo fato de que esses significados eram privados e não documentados, e as tentativas de transformar as planilhas eletrônicas em artefatos compartilhados normalmente falhavam.

Tabela 2. Os paralelos principais entre as cidades e a TI de Pat Helland

Cidades	Lojas de TI
Fábricas ou prédios	Aplicações
Transporte	Comunicação
Bens manufaturados	Dados estruturados
Conjuntos manufaturados	Empresas virtuais
Varejo e distribuição	Processos de negócios
Infra-estrutura urbana	Infra-estrutura de TI
Governo da cidade	Governança de TI

Tabela 3. Implicações de assimetria: os três dilemas

Simetria	Pressuposições	Implicações da assimetria
Tecnologia = Produto	A primeira pressuposição simétrica é que os três primeiros desafios estão todos alinhados. Assim, esses três desafios entram em colapso em uma única dimensão definida pela tecnologia.	Com a SOA, estamos num confronto cada vez maior com as empresas que não são nada além de uma plataforma de tecnologia para outras empresas (da própria Microsoft para baixo). O simples alinhamento não funciona. Em vez disso, elas são empurradas para alguma forma de estratificação.
Negócio = Solução	As regras e procedimentos comerciais sustentados pelo fornecedor se unem às formas como os serviços devem ser usados.	Supõe-se que a manutenção dos trilhos forneça estradas de ferro confiáveis e seguras. No RU, a Network Rail (antigamente Railtrack) pega os serviços de entrada das empresas de engenharia e os transforma em serviços de saída para as empresas de operação dos trens. Ficou comprovado que é extremamente difícil alinhar as exigências de entrada com as exigências de saída.
Demanda do cliente = experiência do cliente	A fantasia do sistema bancário de um processamento direto se baseia na simetria e em valores compartilhados ao longo da cadeia de valor.	Compare esta característica com a situação da indústria farmacêutica, na qual um grupo de relacionamentos de uma empresa de remédios com clínicos gerais e farmacêuticos é de natureza um pouco diferente daquele dos clínicos e farmacêuticos com os seus clientes, a despeito da tendência das empresas de remédios de assumir que deveria ser de outra forma.

É onde entra a fábrica de software. Existe aí uma grande oportunidade de produzir ferramentas de software capazes de suportar uma rica diversidade de demanda do usuário final. Um DSL (Domain Specific Language) pode ser uma maneira de fazer a ligação e manter aberto o intervalo entre o geral/público e o específico de contexto/privado e manter a interação dinâmica entre suprimento e demanda. Essa dinâmica precisa ser conduzida pela maneira como o usuário final define o relacionamento entre os domínios e seus negócios como um todo.

A economia de serviços é um ecossistema complexo. As soluções orientadas para serviços são essencialmente sistemas de sistemas e sua composição deve estar ciente da complexa teoria dos sistemas. Para manter a variedade indispensável (e, portanto, a sobrevivência dos mais aptos) em um ecossistema assim, precisamos de diversidade em todos os níveis de abstração.

Concentração de poder. O sistema econômico da Wal-Mart é insustentável. Ele destrói a estrutura de pequenas lojas das quais uma rica vida urbana depende.

As cidades são de certo modo paradoxais. Por um lado, uma cidade é uma concentração de atividade humana. Mas os processos de concentração são instáveis e podem resultar em formas urbanas altamente anômalas.

Historicamente as cidades possuíam muralhas para impedir a entrada de visitantes indesejados. Em outra parte, Helland defende o modelo de computação de fortaleza. Mas aqui ele parece estar vislumbrando uma estrutura metropolitana contínua, em que uma cidade se une com a próxima (como Manchester se funde com Salford).

Investimento central. A opinião de Helland por investimento central (a posição de Lewis Mumford) oferece uma justificativa para o planejamento central corporativo e o investimento emTI. Muitas corporações grandes e pequenas tentam impor planejamento central de TI. Porém , em muitas organizações essa é uma batalha perdida. A situação real da indústria de TI emerge de milhões de pequenas decisões de compra e está mais perto da idéia de compras anárquicas (a posição de Jane Jacobs).

Salingaros apóia-se em Alexander para descrever como a disputa entre Mumford e Jacobs pode ser resolvida, mas adotando uma abordagem que vai além da simples tentativa de reconciliar o movimento de cima para baixo com o de baixo para cima.

Preservação do "sagrado". Uma versão moderna do sagrado pode ser encontrada na noção de Borgmann de coisas e práticas focais. A função do controle urbano/do sistema seria criar/preservar um espaço no qual essas coisas e práticas locais possam ser desenvolvidas e consideradas.

Em uma economia de demanda ágil, a origem do sagrado é a demanda. Isso contrasta com a lógica do lado do suprimento baseada em uma suposição de demanda simétrica, em que os mercados são definidos para refletir o fornecedor, de modo que as formações de demanda são simétricas às formações de suprimento.

Ao falar sobre Empresas virtuais, Helland diz: *"É necessário considerar o contexto em que a peça será usada. O peso ou a robustez é a preocupação principal?"* Helland argumenta que os padrões são a chave para permitir que os fornecedores de componentes aproveitem o custo da otimização através de um mercado mais amplo e isso pode ser entendido como lógica do lado do suprimento. Essa questão, porém, vai além da padronização (estendida aqui a modelos de componentes de negócios, permitindo que sejam encapsulados como recursos de componentes e orquestrados como uma parte de conjuntos de processos maiores) e abre a granularidade de recursos de componentes uns em relação aos outros. Ou seja, o que é o repertório de recursos de componentes alternativos disponíveis. Isso precisa ser entendido também do lado da demanda.

Ao considerar o Processo de Negócios, esse argumento é estendido por analogia à necessidade de padronização e permutabilidade de dados e operações: *"as pessoas aceitam alegremente coisas padronizadas e a personalização é rara e cara. Mas o processo de negócios ainda é, em grande parte, um trabalho manual. Os padrões são insatisfatórios..."* e assim segue o argumento a favor da padronização, fornecendo uma base para estender uma lógica do lado do suprimento mais profundamente no fornecimento de serviços, com o processo de negócios tornando-se a força impulsora que comanda o formato e a forma das aplicações, "da mesma forma como a Wal-Mart dirige os padrões de inúmeros bens manufaturados."

Isso não enfrenta o desafio do ciclo de varejo acima [5]. Esse ciclo descreve o surgimento de uma nova forma de relacionamento suprimento-demanda (destino) que expande para tornar-se uma nova forma de oferta ao lado de outras (comparação) antes de tornar-se mercadoria (custo) e, no final, incorporar-se no contexto do usuário (personalização). Nesse ponto o terreno está preparado para um novo ciclo (a extremidade de transição) e assim por diante.

Esse ciclo é um processo dinâmico no qual o lado do suprimento está aprendendo constantemente novas formas de suprimento em resposta a uma demanda que está sempre evoluindo e nunca plenamente satisfeita. A demanda assimétrica descreve a demanda no seu contexto de uso específico e esse "algo sempre deixado para ser desejado" é um déficit estrutural que está sempre ali conduzindo o desenvolvimento dos mercados. A transformação em mercadoria é apenas a parte da história do lado do suprimento, sendo a questão real a maneira como a dinâmica da formação da demanda deve ser suportada.

Natureza da Ordem

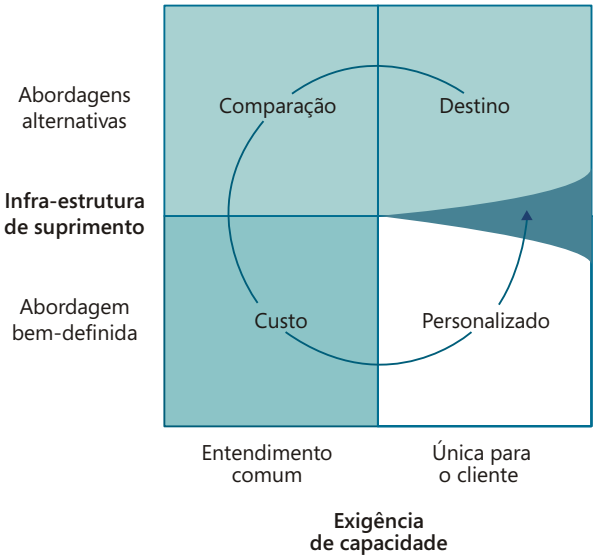
Há várias décadas Christopher Alexander explora alternativas à prática arquitetônica convencional. O seu trabalho mais recente, a Natureza da Ordem, foi publicado no ano passado.

Segundo Alexander, sistemas grandes e complexos não podem ser produzidos por um processo de design convencional, seja de cima para baixo ou de baixo para cima. Em vez disso, eles surgem de um processo estendido e colaborativo (evolucionário). Ordem e coerência resultam das regras que controlam esse processo evolucionário.

Esse processo evolucionário pode ser decomposto em etapas específicas, que podem preservar a estrutura e a totalidade ou destruí-la. A estrutura e totalidade são articuladas como um sistema de centros recorrente.

Os serviços podem facilmente ser considerados como centros de valor [6]. Os serviços podem ser organizados como serviços compostos, com a orquestração dos serviços produzindo (esperançosamente) coerência entre níveis recorrentes.

Figura 1. Requisito de capacidade



Desse modo, uma empresa orientada a serviços pode ser entendida como uma teia de serviços corporativa contínua. As propriedades arquitetônicas dessa empresa dependem dos numerosos processos colaborativos que produzem a sua composição. Se elas forem preservadoras da estrutura, a empresa pode tornar-se cada vez mais diferenciada e cada vez mais integrada, sem perda de coerência.

Alexander é muito crítico do controle convencional sobre o planejamento da cidade e o design urbano e muito crítico da inflexibilidade e dos resultados desumanos da composição dirigida. Acreditamos que as suas idéias sobre design e ordem são consistentes com as necessidades da composição colaborativa, como discutido neste ensaio.

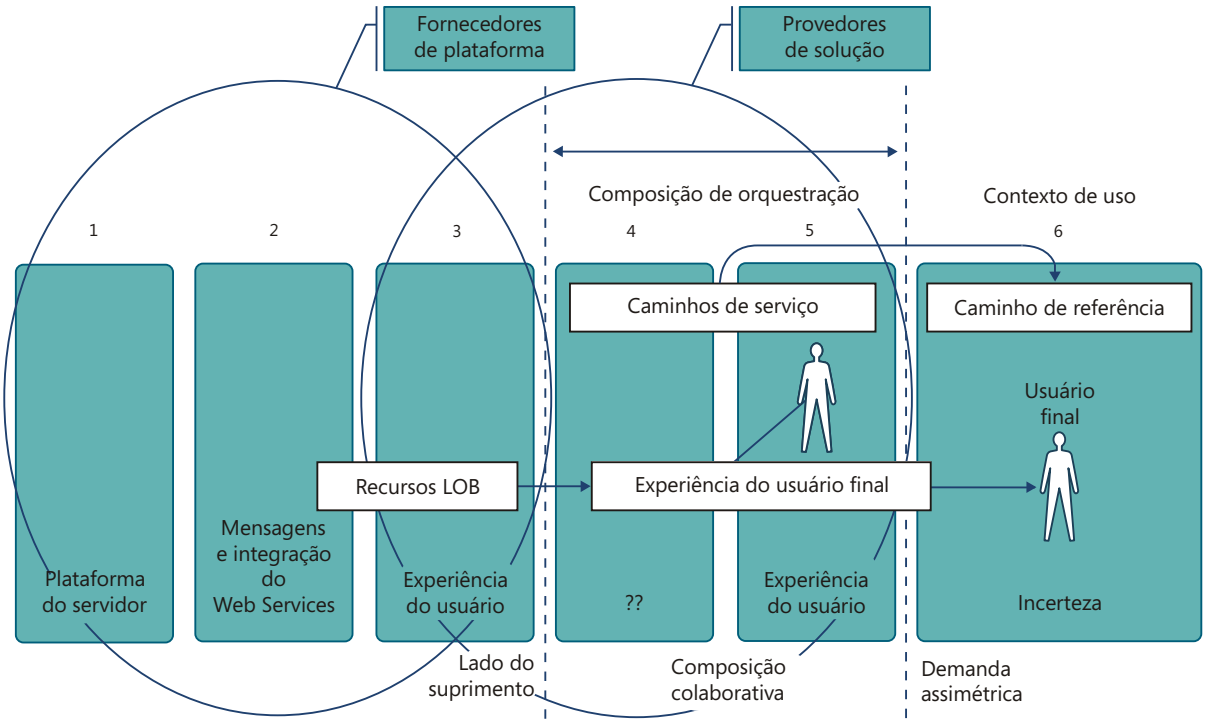
Implicações estruturais

Uma cadeia comercial ou de valores é composta em uma estrutura geométrica. Na SOA, criamos uma cadeia comercial ou de valores como uma rede de serviços. Esse é um poderoso padrão geométrico. Mas pode haver muitas geometrias de rede possíveis capazes de satisfazer um requisito comercial determinado, todas elas satisfazendo os princípios da SOA. Por exemplo: hub/spoke ou peer/peer.

Uma característica importante da SOA é a **estratificação**. Um processo comercial é composto de serviços de um conjunto de serviços de nível inferior, apresentado como uma plataforma. Um bom exemplo de uma plataforma de negócios é o conjunto de serviços de varejo oferecido pela Amazon e pela eBay. Outros fornecedores de serviços construíram serviços logísticos/de varejo em cima das plataformas Amazon/eBay.

Cada plataforma, por sua vez, é construída sobre serviços mais inferiores ainda. Nos níveis inferiores pode haver coleções de serviços baseados em TI, conhecidas como ESB (Enterprise Service Bus). Também pode haver plataformas de serviços sócio-técnicas como os centros de chamadas. Algumas dessas camadas inferiores da pilha podem parecer serviços puramente técnicos; no entanto, uma observação mais completa revela a existência de uma organização de TI que mantém a plataforma; em outras palavras, é também uma plataforma sócio-técnica que inclui seus administradores e programadores.

Figura 2. Composição colaborativa na cadeia de valores de seguros baseada em Microsoft IVC [7] (Fonte: CBDI Forum, Novembro de 2004)



Assim, temos uma geometria estratificada na qual uma pessoa que enfrenta um problema em um nível determinado recebe um conjunto de serviços disponíveis, transformados em uma plataforma virtual. Isso pode ser considerado como uma **pilha comercial**, com uma plataforma empilhada em cima de outra. E embora os princípios da SOA possam oferecer certa orientação geométrica e validar determinados padrões geométricos, ainda existe um trabalho de design para determinar a forma de geometria mais apropriada para suportar o serviço em demanda. Esse trabalho de design pode ser fácil quando o requisito é trivial, mas fica mais difícil à medida que a complexidade aumenta.

Em muitas situações, o lado da demanda tem mais variação do que um designer humano (ou equipe de design) pode acomodar. (Isso nós caracterizamos como uma assimetria da demanda, o que pede um processo de design assimétrico.) Nessas circunstâncias, precisamos ir mais adiante ainda e começar a pensar sobre soluções de geometria variável, em que a própria geometria pode ser adaptada sob demanda.

Por exemplo, no passado considerávamos que a granularidade tinha de ser fixada no momento do design. Mas podemos conceber uma plataforma de Web Services que detecta padrões de composição do lado da demanda, define novos serviços componentes automaticamente, descrevendo e publicando esses novos serviços em tempo real, e notifica possíveis usuários do novo serviço, completa com um incentivo para alternar para novas maneiras de orquestrá-los em suporte da composição do lado da demanda. Podemos conceber essa plataforma de Web Services analisando o conteúdo das mensagens de um serviço determinado e produzindo um serviço substituto com uma base menor que satisfaça a maior parte dos usuários de uma maneira mais elegante.

Utilizamos o termo **paisagem de valor** para nos referirmos à distribuição de custo, benefício e know-how através de um ecossistema de mercado complexo, como a indústria de seguros, para um nível de risco determinado. A tecnologia (incluindo a SOA) influência a geometria comercial porque afeta não somente os custos da transação, mas também a maneira como o know-how pode ser aproveitado em relação à demanda. O formato da paisagem de valor muda (já começou a mudar) como resultado de B2B, B2C, P2P e BPO. Empresas que antes ocupavam posições de mercado seguras podem descobrir que a sua vantagem comercial está se desvanecendo ou que se encontram desligadas de seus antigos clientes ou cadeias de suprimento.

Identificar Objetivos

Vamos supor que uma empresa de seguros tenha as seguintes metas estratégicas:

- **Lucratividade, viabilidade a curto prazo.** Fornecer o máximo de valor de serviços da maneira mais econômica possível, utilizando as tecnologias e os serviços de entrada disponíveis da maneira mais eficiente possível, com o mínimo de custos/riscos de mudança.
- **Adaptabilidade, viabilidade a médio prazo.** Entender e reagir à demanda em transformação dos serviços de seguros e às tendências de custo e risco, tanto internamente como através do segmento de mercado. Desenvolver e implantar novos serviços para explorar novas oportunidades comerciais e evitar ameaças comerciais emergentes.
- **Sobrevivência, viabilidade a longo prazo.** Assegurar que a proposição comercial principal permaneça válida e não seja desgastada por participantes mais ágeis. Tomar ação estratégica em relação às transformações estruturais na indústria de seguros.

Se estivermos fazendo geometria comercial para uma companhia de seguros, precisamos pensar sobre a indústria de seguros como um ecossistema em evolução. Precisamos de um modelo COMO ESTÁ do ecossistema atual (em grande parte baseado em tecnologias pré-SOA) e um modelo A SER de um ecossistema emergente (baseado nos

efeitos da SOA). Podemos esperar que o ecossistema pré-SOA evolua para alguma forma de ecossistema pós-SOA, embora não tenhamos muita idéia de qual das transformações possíveis vai ocorrer primeiro. Para satisfazer todas as três metas estratégicas, uma companhia de seguros precisa explorar o ecossistema pré-SOA e preparar-se para o ecossistema pós-SOA.

Observe que essa situação pode forçar a companhia de seguros a implementar uma geometria variável através da sua pilha comercial, tanto na plataforma organizacional quanto na plataforma de TI de base. Caso contrário, ela terá que operar abaixo do modo mais eficiente durante um período prolongado ou incorrer em custos de organização e de TI substanciais toda vez que o mercado der um outro passo em direção à SOA. A geometria variável envolve uma colaboração dinâmica ***(composição colaborativa)*** entre um lado de suprimento eficiente e um lado de demanda variável (assimétrica) como mostrado na figura 2 [7].

Assimetria da demanda

Assimetria significa que as formas de demanda são cada vez mais específicas ao contexto em que surgem.

A primeira assimetria envolve separar tecnologia do suprimento de produtos específicos.

Isso requer a modelagem de comportamentos possíveis que podem ser suportados (de modo que a Microsoft ou a indústria de automóveis deve se modularizar em suporte de famílias de uso de tecnologia).

A segunda assimetria requer a separação de modelos comerciais que podem organizar suprimento a partir das soluções que estão em oferta. Isso requer modelagem das formas possíveis de geometria comercial (assim, manutenção de trilhos ou serviços de varejo devem usar um modelo de franquia para permitir que a variação na organização comercial acomode a variedade de maneiras em que o serviço precisa ser implementado).

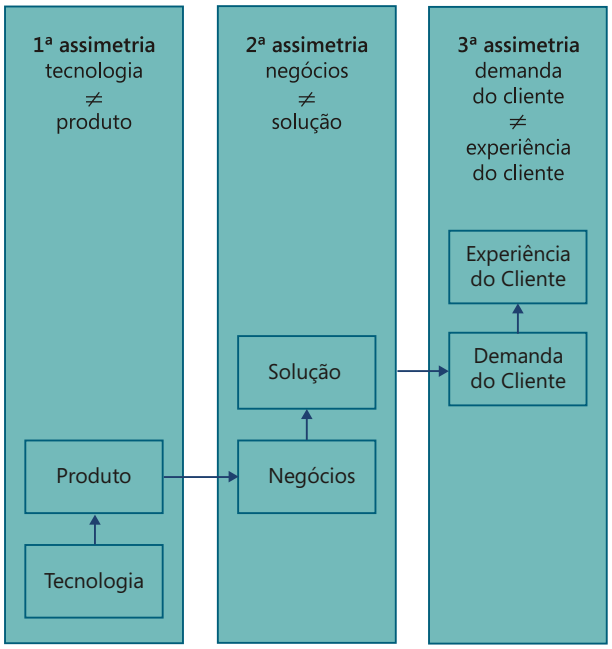
E a terceira assimetria requer a separação dos diferentes contextos de uso. Isso requer modelagem das formas de demanda possíveis (de modo que o serviços financeiros ou de atendimento estão tendo de adotar a maneira como os bens/condições são gerenciados ao longo do tempo de modo que responda às diferentes formas de contexto de uso).

Essas assimetrias estão resumidas na Figura 3 e vale a pena considerar o que acontece se forem ignoradas. Na primeira assimetria, isso significa definir o produto pela tecnologia. Isso é típico dos estágios iniciais do surgimento de novas tecnologias. (Lembra-se de como costumávamos precisar usar telefones celulares?) Na segunda assimetria, isso significa definir a solução para o cliente pela maneira como os negócios são organizados. (Lembra-se como as grandes empresas costumavam relacionar-se com os clientes antes do CRM (Customer Relationship Management)?). E na terceira assimetria, a solução do problema apresentada pelo cliente é considerada como sendo o que o cliente realmente precisa. (Você já recebeu uma receita do médico que no final trata somente o sintoma?). Consideramos que o principal impacto competitivo da SOA é que ela altera o que o fornecedor pode dar-se ao luxo de ignorar a partir da perspectiva do cliente.

Para melhorar a captura de formas de demanda assimétricas, uma organização precisa de liderança que permita que ela faça duas coisas:

- **Levar poder à extremidade da organização:** As pessoas na extremidade da organização com o relacionamento com a demanda assimétrica devem ser capazes de organizar o modelo comercial que necessitam para capturar essa demanda.

Figura 3. Três assimetrias de demanda



- **Desenvolver uma infra-estrutura ágil:** fornecendo serviços comerciais que possam ser orquestrados e compostos na extremidade em resposta às formas de demanda específicas que estão objetivando. Isso permite que o lado do suprimento de uma empresa extraia economias de escala ou escopo ao fornecer suporte através de vários modelos comerciais.

Exemplo da assistência médica.

A assimetria na variedade de condições que as pessoas possuem e nas maneiras como ela se revela nas vidas das pessoas ao longo do tempo está crescendo cada vez mais. Enquanto isso, hospitais e clínicas estão precisando tornar-se cada vez mais eficientes na maneira como administram caminhos de atendimento específicos.

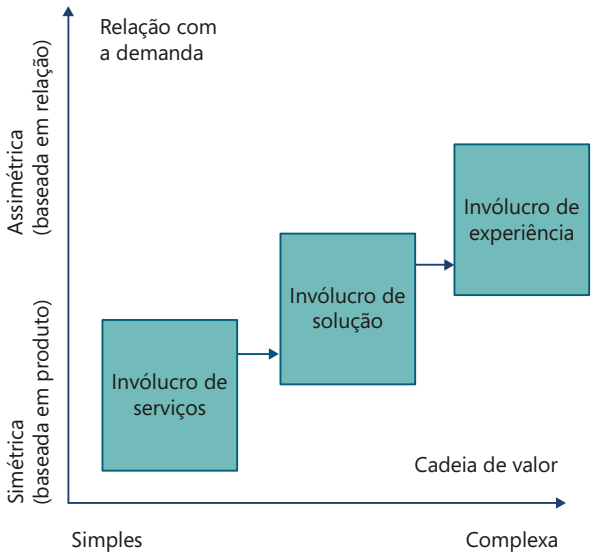
Observamos esse desafio particularmente em relação a condições crônicas. Esses pacientes não produzem condições que se encaixam nos tratamentos disponíveis, enquanto que organizar sistemas de tratamento agudo para o tratamento de condições crônicas tem um custo exorbitante. As condições dos pacientes são implacavelmente assimétricas do ponto de vista dos especialistas médicos que estão tentando cuidar delas. Por exemplo, uma cirurgia aguda geralmente pode ser rastreada até uma falha anterior em fornecer tratamento profilático na hora certa.

Resposta estratégica

Levar poder à extremidade na assistência médica significa os recursos financeiros seguindo o paciente e os médicos tendo a capacidade de criar um plano de tratamento que seja específico à condição do paciente.

Isso cria um duplo desafio para o fornecimento de assistência médica. Não somente deve ser aumentada a flexibilidade com que seus serviços componentes podem ser tornados disponíveis aos pacientes, mas também os médicos precisam ter um envolvimento muito maior na formulação de estratégias de fornecimento de assistência médica que possam ser personalizadas para a condição de um paciente ao longo do tempo e por cujo fornecimento eles possam ser considerados responsáveis. Onde isso puder ser feito, o custo total da assistência é reduzido.

Figura 4. A estratégia da oferta/demanda



Uma PASA (agência de compras e suprimentos de assistência médica) era responsável pelo suprimento de equipamento a um serviço clínico. Eles estavam preocupados com os efeitos colaterais de minimizar os custos desses suprimentos: investimento reduzido na indústria e um círculo vicioso de declínio na qualidade do próprio serviço clínico. Decidiram que o design simétrico convencional não estava funcionando para eles; em uma tentativa de aprimorar a qualidade do serviço clínico, decidiram considerar o desenvolvimento de uma abordagem que tratasse da natureza assimétrica das demandas da clínica. Eles conduziram um estudo piloto inicial para estabelecer a viabilidade de um processo de design assimétrico.

O que PASA aceitou foi que precisavam tratar do lado da demanda da clínica e estabelecer a melhor maneira de satisfazer as suas necessidades. Dentro desse contexto de uso, poderiam tratar a questão dos custos do suprimento.

Foi estabelecido um processo nacional pela agência de modernização. Esse processo realizou seis projetos pioneiros, cada um deles direcionado para estabelecer como realizar mudanças em cada contexto. Foi tarefa do projeto nacional garantir suporte e financiamento a longo prazo para o processo de mudança à luz do aprendizado e dos resultados dos projetos pioneiros. No final, isso evoluiu para a modelagem do impacto regional e nacional das mudanças nos orçamentos do Sistema Nacional de Saúde e dos Serviços Sociais.

Do ponto de vista do suprimento para as clínicas, a modelagem do lado da demanda foi dos caminhos de orientação e dos serviços oferecidos por cada clínica em resposta às demandas que surgiram desses caminhos de orientação. A modelagem do lado do suprimento foi da organização da própria clínica, junto com o uso de fornecedores, de forma a estabelecer como um estava alinhado ao outro. Onde esse "corte" ocorreu entre o lado do suprimento e o lado da demanda foi função de quem era o cliente e qual era a sua programação de mudança. Não obstante, ao examinar os caminhos de orientação e as maneiras específicas em que eles próprios tinham sido ***"colonizados"*** por fornecedores, outras questões foram levantadas sobre a organização do próprio atendimento primário. Essas questões, no entanto, foram deixadas para serem abordadas por um sistema de cliente diferente em um momento posterior, o qual teria que tratar dos interesses das Autoridades de Saúde Estratégica.

O desafio principal era dar aos clínicos "controle de design" sobre a maneira como as clínicas operavam (ou seja, poder na extremidade). Fundamental para isso era compreender, no lado do suprimento, a natureza crônica de vários episódios das condições que estão sendo tratadas pela clínica e, no lado da demanda, os processos de delegação e/ou assumir em equipe a responsabilidade clínica pelas condições dos pacientes. Os clínicos não possuíam os meios de definir as diferentes características do primeiro e gerenciar as complexidades do segundo . Além disso, sem os meios de realizar essas coisas, não havia uma maneira prática de tornar os clínicos responsáveis pelo desempenho clínico da clínica. A solução era construir uma plataforma de relatórios que pudesse fornecer suporte à realização dessas coisas.

Essa modelagem envolvia definir os caminhos de orientação e suas características reais. Disso surgiu o requisito de alterar a maneira como a clínica estava se relacionando com a demanda.

Essa modelagem envolvia definir as proposições de serviços e os modelos comerciais clínicos necessários. No primeiro caso, isso significava estabelecer protocolos de episódio para diferentes condições e, o segundo, alterar os processos de fluxo de trabalho entre a clínica e os processos de atendimento e administração em suporte. Não foi tentado o realinhamento das infra-estruturas de suprimento, ganhos consideráveis sendo disponíveis simplesmente através da maneira como foi gerenciado o alinhamento dos fornecedores com a demanda. A plataforma de relatórios construída forneceu os meios de conseguir isso. A plataforma permitiu à clínica definir seus próprios protocolos de tratamento em relação às suas próprias definições de condições de orientação. O back end da plataforma conseguiu levantar dados do ambiente do Sistema Nacional de Saúde sobre pacientes, consultas etc .

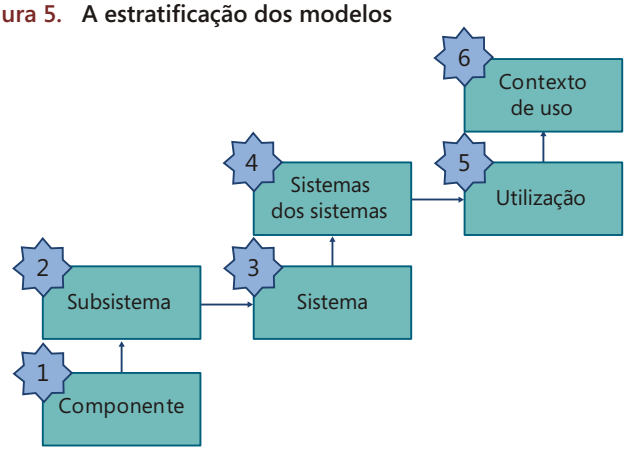
Esquema de suprimento-demanda

A Figura 4 mostra um esquema típico da empresa baseada em serviços à medida que ela responde ao impacto competitivo da SOA. A empresa enfrenta uma forma de assimetria de cada vez:

1. Utiliza um "invólucro de serviços" para separar o produto da tecnologia. Isso inclui definir um modelo de objeto diferente para o lado da demanda, separando dados "não processados" do que poderíamos chamar dados "preparados";

2. Utiliza um "invólucro de soluções" para separar a solução da empresa. Isso inclui definir regras diferentes para o lado da demanda, separando a lógica comercial da orquestração de diferentes soluções;

Figura 5. A estratificação dos modelos



3. Utiliza um "invólucro de experiências" para separar a experiência contínua do cliente das soluções específicas compradas pelo cliente em algum momento. Isso inclui novas formas de modelagem de processo para entender a experiência de soluções do cliente dentro de seus contextos de uso específicos.

A questão sobre essa progressão é que ela confronta as empresas de suprimento com a necessidade de gerenciar uma complexidade crescente (e simultaneidade) na maneira como a cadeia de valores se relaciona com o cliente. Daí a importância competitiva da SOA.

Quebrando as três simetrias

Com a primeira simetria, a empresa precisa de apenas um modelo, porque a demanda pode ser deduzida do suprimento (ou assim parece no momento!). Quando essa simetria é quebrada, dois modelos são necessários: um para gerenciar a tecnologia e outro para gerenciar os negócios. Quantas vezes um investidor de capital de risco teve de ensinar isso a uma empresa iniciante?

Se o pleno potencial de uma proposição comercial deve ser concretizado, a segunda simetria também deve ser quebrada (isso é coisa que todo mestre em administração de empresas aprende). Agora há três modelos: um para gerenciar a tecnologia, um para gerenciar os negócios e um para gerenciar a solução de mercado. Assim, no exemplo da estrada de ferro na Caixa 6, é necessário diferenciar a maneira como o serviço é fornecido porque não é possível deduzir os níveis de serviço do lado da saída das empresas que operam o trem puramente dos requisitos de serviço do lado da entrada da infra-estrutura da estrada de ferro (a menos, claro, que você esteja preparado para fornecer serviço no mais alto nível de qualidade sob todas as circunstâncias).

Mas qual o efeito de quebrar a terceira simetria? O que o CRM (Customer Relationship Management) nos diz é que a "solução de mercado" é também um relacionamento com o cliente. Mas ao quebrar a terceira simetria devemos aprender que cada cliente incorpora a solução em um contexto de uso diferente. Assim, na assistência médica não é mais apenas uma questão de fornecer tratamentos específicos de forma econômica, mas também de fornecer o tratamento correto na hora certa dentro do contexto de uma condição do paciente em evolução, como idade avançada, por exemplo!

No caso farmacêutico, não é possível extrair o conhecimento do lado da saída (o que os médicos clínicos e os farmacêuticos precisam saber sobre a condição do paciente) do conhecimento do lado da entrada. São necessários pelo menos dois modelos porque não é possível deduzir os efeitos dos tratamentos puramente do conhecimento das interações entre a prescrição e a situação que se apresenta no momento da prescrição; é necessário um modelo dos efeitos dinâmicos da prescrição dentro do seu contexto de uso.

No total, a quebra das simetrias requer que seis níveis de modelo diferentes sejam separados (observe a Figura 5). Embora cada um desses extratos possa conter muitos níveis diferentes, a agilidade da infra-estrutura dependerá de quanta variabilidade eles permitirão na geometria da qual eles podem ser compostos.

Função do arquiteto de TI na SOA

O que isso tudo requer do "arquiteto de TI" em suporte dessas novas formas de agilidade? O nosso argumento principal é que não existe um nível correto único de granularidade. Existe um dilema de lado do suprimento versus lado da demanda que precisa ser abordado por um método orgânico (Cf. Alexander) no qual o nível de estratificação em que uma mudança é feita também deve sempre abordar o seu impacto nos níveis superior e inferior. Como resultado, uma troca deve ser feita entre dois tipos de pressão em qualquer camada de estratificação determinada, uma para padronização e a outra requerendo maior diversidade. Isso coloca um tipo diferente de demanda na função da arquitetura.

Podemos pensar na função da arquitetura como a formação de determinados tipos de julgamento (estrutural) [8]. Consideramos a arquitetura como algo que surge de um processo colaborativo, que pode (mas não precisa) envolver pessoas que tenham a palavra "arquiteto" no nome do cargo. Podemos nos referir a essas pessoas como arquitetos profissionais, embora na engenharia de software e de sistemas não exista uma corpo profissional que conceda algum status oficial a esse cargo. Esses julgamentos pertencem a um regime de design assimétrico.

Podemos assim separar os produtos e processos da atividade arquitetônica das responsabilidades profissionais de Arquitetos devidamente qualificados. As propriedades estruturais de uma cidade dependem de um processo colaborativo complexo no qual os julgamentos profissionais dos arquitetos são colocados frente a uma enorme variedade de interesses comerciais e políticos. Os arquitetos profissionais raramente dominam o design de cidades, embora possam às vezes exercer alguma influência.

E podemos observar uma situação semelhante na TI. Embora haja muitas pessoas na indústria da TI que se autodenominam "arquiteto", as propriedades estruturais dos grandes sistemas de TI dependem em grande parte de decisões tomadas em outras partes. Por exemplo, o grau de ligação entre dois módulos pode ter um impacto significativo na coesão estrutural e na flexibilidade de um sistema grande, mas isso pode ser determinado por opções de desenvolvimento em nível razoavelmente baixo que não são nem mesmo visíveis aos arquitetos de software.

Em muitas organizações, os arquitetos de software podem ser manobrados e marginalizados por coalizões de desenvolvedores e usuários. E à medida que aumenta a terceirização, a posição do arquiteto de software pode tornar-se mais fraca ainda, particularmente nos casos em que as especificações contratuais concentram-se nos requisitos funcionais e especificam de forma insuficiente as propriedades estruturais; e onde existem mecanismos inadequados para os arquitetos verificarem as propriedades estruturais do software fornecido. (Por exemplo, ligações ocultas que comprometem a flexibilidade pretendida de um artefato de software.)

Desse modo, o desafio dos arquitetos de software é permanecerem relevantes em um mundo de SOA, em um mundo de produção distribuída de serviços distribuídos, prestando atenção aos problemas estruturais reais que surgem nesse mundo "sob demanda". Caso contrário, serão incapazes de contribuir com algo de valor ao design e gerenciamento de sistemas sob demanda de sistemas. Isso leva a uma necessidade de formas de análise que oferecem suporte a um regime de design assimétrico e podem permitir que seja dada uma consideração explícita às opções implícitas que estão sendo feitas em relação à geometria.

Conclusões e próximas etapas

Temos visto alguns fornecedores reconhecerem os problemas do lado do suprimento de reconciliar vários Web Services (IBM Rational, por exemplo), enquanto outros fornecedores de plataformas estão criando as condições para uma explosão na quantidade domínios (comportamentais) que necessitam ser trazidos uns em relação aos outros (Microsoft, por exemplo). Nos dois casos, a empresa orientada a serviços é configurada como uma estrutura de serviços contínua "a teia corporativa". Mas isso nunca pode ser conseguido em um projeto grande e ambicioso. Deve ser obtido de forma progressiva através de um fluxo contínuo de projetos pequenos e médios.

Na abordagem do planejamento colaborativo, ordem e coerência emergem da atividade distribuída, sem qualquer autoridade central de design. Cada unidade de compras, desenvolvimento ou manutenção deve ser considerada como um projeto, com as saídas do projeto sendo constituídas como serviços. Desse modo, cada projeto contribui com algo positivo para a teia de serviços corporativa emergente. Assim, que forma de controle é necessária para manter a ordem da arquitetura?

Controle da SOA é requerido para garantir que cada projeto satisfaça às demandas globais da teia corporativa e para garantir que haja uma mistura de projetos bem equilibrada, tipos diferentes além de escalas diferentes (grande, média e pequena).

O que a nossa discussão na Parte I descreveu são as limitações de uma abordagem da linha de suprimento para o controle da SOA; a composição dirigida é limitada na sua capacidade de responder à heterogeneidade completa da demanda. Ela deixa um déficit de valor muito grande em relação à demanda que é cada vez mais heterogênea, assimétrica e diferenciada no espaço e também no tempo. Portanto, precisamos levar o controle à extremidade da organização, reconhecendo que estamos engajados em processos de design assimétrico, e nos prepararmos para atender o século XXI em seus próprios termos. Na Parte II deste artigo iremos levantar a questão de o que levar o controle "à extremidade" significa para o design de infra-estruturas da SOA e também para a relação com a demanda.

Referências

Christopher Alexander, The Nature of Order.4 Volumes. The Center for Environmental Structure, 2002-2004.

Albert Borgmann, Technology and the Character of Contemporary Life. Chicago, 1984.

Philip Boxer & Bernie Cohen, Triple Articulation. BRL Working Paper, revised 2004.

Andrew Coward & Nikos Salingaros, The Information Architecture of Cities, Journal of Information Science, Volume 30 No. 2 (2004), pp. 107-118. Reeditado em Salingaros 2005.

Jack Greenfield & Keith Short, Software Factories (Wiley, 2004).

Pat Helland, Metropolis. Microsoft Architects Journal [2], April 2004. Disponível em *http://msdn.microsoft.com/architecture/journ/*

Jane Jacobs, The Death and Life of Great American Cities (Vintage Books, New York, 1961).

Lewis Mumford, The Culture of Cities. (Secker & Warburg, 1938).

Richard C. Murphy, Centers: The Architecture of Services and the Phenomenon of Life. FTP Online March 2004. *http://www.ftponline.com/special/soa/murphy/*

Nikos Salingaros, Principles of Urban Structure. Delft University Press, Delft, Holland, 2005, no prelo. *http://www.math.utsa.edu/~salingar/urbanstructure.html*

Richard Veryard, Component-Based Business (Springer 2001).

Richard Veryard & David Sprott, The Service Based Business (CBDI Journal, 2003). SOA Governance and Business Driven SOA (CBDI Journal, 2004).

Agradecemos a Bernie Cohen, Pat Helland e Nikos Salingaros pelos seus comentários sobre os rascunhos iniciais.

Notas de rodapé

1. Para obter um resumo útil, consulte Coward & Salingaros.

2. Pat Helland, Metropolis. Microsoft Architects Journal [2], April 2004. Disponível em *http://msdn.microsoft.com/architecture/journ*

3. Exemplo prático: as empresas telefônicas gostariam de considerar a localização das antenas dos telefones celulares como mera infra-estrutura a ser decidida em bases técnicas, sem qualquer consulta pública. No entanto, as pessoas estão se preocupando com a radiação dessas antenas, particularmente perto de residências e escolas, e isso torna política a localização das antenas.

4. Uma possível reconciliação da transformação dos valores humanos em mercadoria é fornecida por Albert Borgmann.

5. Para obter uma versão desse ciclo na sua aplicação à assistência médica, consulte "Triply Articulated Modelling of the Anticipatory Enterprise" por P Boxer e Prof B. Cohen, Proceedings of the International Conference on Complex Systems, Boston 2004.

6. Consulte o artigo de Murphy.

7. Imagem do relatório sobre Service-Based Business in Insurance, CBDI Forum Novembro de 2004.

8. Esse exemplo é resultado de trabalho feito para PASA. *http://www.pasa.nhs.uk/orthotics/Orthotic_Pathfinder_Report_July_2004.doc*

9. Utilizamos julgamento aqui como definido por Vickers, para incluir apreciação e julgamento de valor, além de julgamento de ação.

Sobre os autores

Richard Veryard é escritor, consultor de gerenciamento e analista de tecnologia em Londres, Inglaterra.

Philip Boxer é consultor estratégico na Inglaterra. Os autores agradecem os comentários de Bernie Cohen, Patt Helland e Nikos Salingaros sobre os primeiros rascunhos.

Integração Baseada em Serviços

Anna Lui e Ian Gordon

Introdução

Este artigo descreve uma abordagem comprovada para auxiliar os arquitetos a avaliar tecnologias de integração de aplicativos corporativos. Em particular, concentrarmos a nossa discussão na avaliação de tecnologias de integração para a implementação de integração baseada em serviços.

A construção de uma solução de integração de aplicativos corporativos é difícil. Essas soluções precisam integrar vários sistemas comerciais que não foram criados para trabalhar juntos. A integração de sistemas assim é difícil por muitas razões. Elas incluem a heterogeneidade das plataformas e das linguagens de programação, a diversidade e complexidade de cada sistema de negócio individual e a dificuldade de entender os requisitos da solução integrada resultante. Os arquitetos de software realizam uma série de tarefas cruciais durante o design de aplicativos corporativos integrados. Entre elas:

- Ajudar a entender os requisitos funcionais e de qualidade dos aplicativos integrados.
- Criar o projeto de arquitetura inicial dos aplicativos integrados.
- Selecionar tecnologias de integração adequadas que possam atender os requisitos dos aplicativos.
- Validar que a combinação da arquitetura e a tecnologia de integração usada para construir o aplicativo de nível corporativo provavelmente terão êxito antes que um grande investimento de implementação seja feito.

Este artigo descreve uma abordagem comprovada para auxiliar os arquitetos a avaliar tecnologias de integração de aplicativos corporativos. Em particular, concentrarmos a nossa discussão na avaliação de tecnologias de integração para a implementação de integração baseada em serviços.

SOA para integração

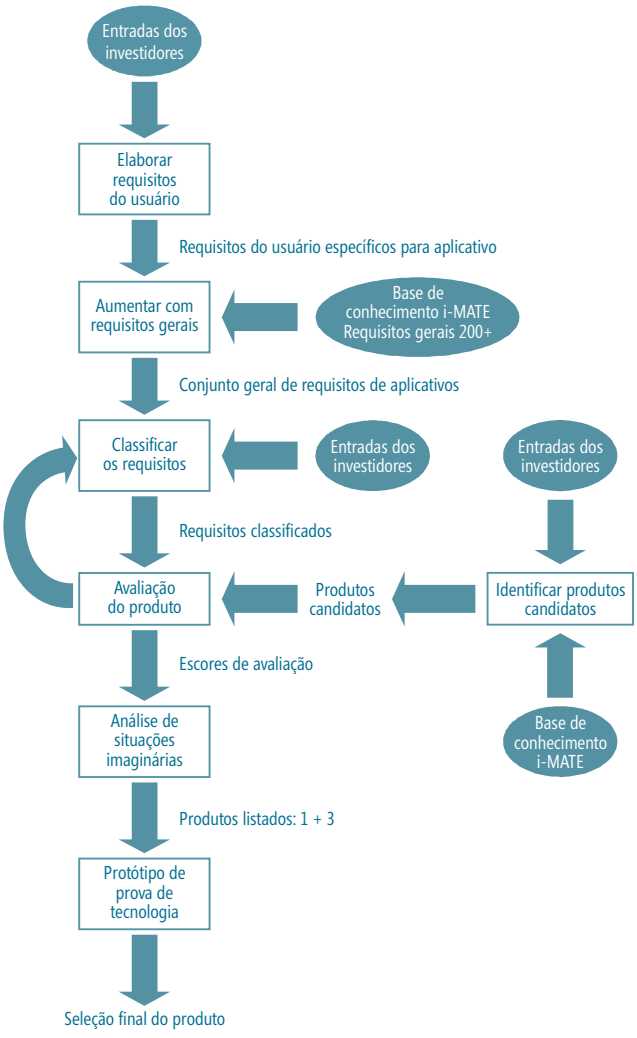
Com o advento de padrões de mercado como Web Services, a Arquitetura Orientada a Serviços está conduzindo uma mudança de paradigma em muitas áreas, incluindo integração de aplicativos corporativos.

A abordagem baseada em serviços para fazer integração refere-se a integrar entidades de computação usando interações de serviços. A abordagem de integração baseada em serviços trata de problemas como a integração de sistemaslegado e heterogêneos inflexíveis, aopermitir que as organizações de TI ofereçam a funcionalidade dos aplicativos existentes como serviços reutilizáveis.

Em contraste com a EAI (Enterprise Application Integration) tradicional, as características significativas da abordagem de integração baseada em serviços são:

- **Interfaces padronizadas e bem definidas** - Os clientes recebem acesso consistente e fácil de entender aos serviços.
- **Opacidade** - A tecnologia e a localização do aplicativo que fornece a funcionalidade estão ocultas atrás da interface de serviço. Na verdade, não existe a necessidade de um fornecedor de serviços fixo.
- **Flexibilidade** - Tanto os fornecedores de serviços quanto os consumidores de serviços podem mudar - a descrição do serviço é a única constante. Enquanto o fornecedor e o consumidor continuarem a respeitar a descrição de serviço, os aplicativos continuarão a funcionar.

Figura 1. Processo de avaliação



Tecnologias para implementar a integração baseada em serviços

As tecnologias para construir a integração baseada em serviços precisam possuir as seguintes funcionalidades básicas:

Entrega de mensagens; Roteamento inteligente; Serviços de eventos; Adaptadores de aplicativos; Transformação de dados/conversão XML; Processamento de regras; Suporte a Web Services; Orquestração de serviço/processo; Gerenciamento de processo de negócio; Monitoramento de atividade de negócio.

Além disso, para garantir o êxito da integração baseada em serviços, a tecnologia de integração precisa possuir as seguintes qualidades:

Escalabilidade; Alto desempenho; Segurança; Gerenciabilidade

Como podemos ver, um dos focos principais é na utilização de padrões do mercado como Web Services para permitir a entrega real de mensagens e diversos outros serviços avançados, evitando os problemas das tecnologias EAI tradicionais, ou seja, o uso de protocolos patenteados para trocas de mensagens. Dessa maneira, a integração baseada em serviços é um padrão de design que garante interoperabilidade e integração real em qualquer paisagem corporativa heterogênea.

Avaliando as tecnologias de integração

Existem muitas implementações de tecnologias de integração que fornecem as funcionalidades citadas acima. Elas variam das tecnologias EAI tradicionais com um complemento com recursos de Web Services a novas implementações com suporte inerente a Web Services.

Infelizmente , selecionar uma implementação de tecnologia de integração apropriada não é uma proposição simples para a maioria das organizações de TI. Existem numerosas razões para isso, mas elas normalmente situam-se em torno de:

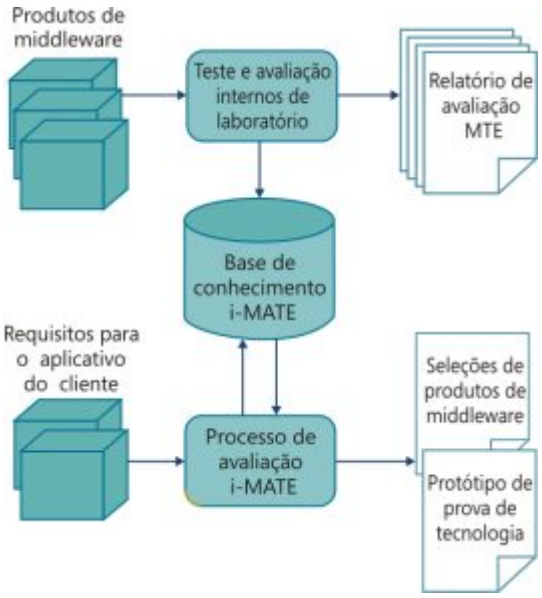
- **Complexidade da tecnologia -** Os produtos de integração são grandes, diversos e possum literalmente milhares de recursos e interfaces de programação de aplicativos. São complexos para entender e os detalhes de baixo nível podem produzir efeitos sérios na maneira como um produto se comporta. O demônio realmente está nos detalhes.
- **Diferenciação do produto** - Existem dezenas de milhares de produtos competindo na arena da integração. Em um nível superficial, muitos possuem conjuntos de recursos e capacidades quase idênticos. A diferença de preços às vezes pode ser bem grande, o que complica mais ainda os problemas de seleção e aquisição.
- **Conhecimento da organização de TI** - As organizações de usuários finais raramente possuem arquitetos e engenheiros com o entendimento amplo e profundo necessário de todas as tecnologias e produtos de integração. Portanto, é um exercício caro e demorado para a organização adquirir esse conhecimento para escolher um produto de integração apropriado. E também distrai a equipe de engenharia principal da sua tarefa básica focada em aplicativos.

O processo i-MATE

i-MATE (Middleware Architecture and Technology Evaluation in Internet time) é um processo especializado de engenharia de software para avaliar middleware COTS (Component Off The Shelf) . É apropriado para organizações que operam no Nível 3 do Software Acquisition Capability Maturity Model do Software Engineering Institute [1], particularmente nos seu suporte das áreas principais de processo Requisitos do Usuário e Gerenciamento de Risco de Aquisição. A eficácia e a inovação do i-MATE devem-se à combinação de:

- **Um processo definido** - Isso compreende uma série direta de etapas de processo bem definidas para reunir, classificar e ponderar os requisitos de aplicativo do middleware de integração.

Figura 2. Preenchendo a base de conhecimento



- **Uma base de conhecimento** - Isso contém várias centenas de requisitos genéricos para diversas classes de produtos de middleware COTS, incluindo os exclusivos de implementações de integração baseadas em serviços.
- **Uma ferramenta de análise de requisitos** - A ferramenta de análise permite uma rápida avaliação, experimentação e apresentação da maneira como os produtos de middleware em avaliação se comparam com os requisitos de projeto.

As seções a seguir descrevem os recursos exclusivos do i-MATE que o tornam altamente adequado para avaliar tecnologias de integração no contexto de construir uma solução de integração baseada em serviços.

O processo usado no i-MATE é semelhante aos descritos em [2,3]. Ele define uma série de etapas que são realizadas no i-MATE e as decisões tomadas e os artefatos produzidos em cada estágio. Elas estão representadas na Figura 1 e são explicadas brevemente aqui:

- **Elaborar os requisitos do cliente** - Essa primeira etapa produz um documento que captura os requisitos do cliente. Como a tecnologia e os problemas do aplicativo são complexos, geralmente consideramos que os requisitos totais não são entendidos inteiramente. Como conseqüência, uma série de workshops são realizadas com os requisitantes do aplicativo para esclarecer os requisitos. Os requisitantes envolvidos idealmente incluem grupos comerciais e de TI. O documento resultante detalha os requisitos técnicos e comerciais que são específicos à necessidade da tecnologia de integração nesse ambiente de integração baseado em serviços. Cada requisito é expresso como um item único que pode ser avaliado em relação a uma tecnologia de integração especifica.
- **Aumentar com requisitos genéricos** - Isso introduz a base de conhecimento do i-MATE de mais de 200 requisitos genéricos, aplicáveis de modo amplo nas tecnologias de integração. Eles aumentam o conjunto de requisitos específicos do aplicativo com requisitos de integração genéricos. O resultado dessa etapa representa os requisitos totais do aplicativo para tecnologia de integração baseada em serviços, representados como pontos de requisito identificados individualmente.

Requerimentos Específicos - SOI

O produto deve dar suporte ao conjunto de especificações do Perfil Básico de Web Services (p.ex., WSDL, SOAP, UDDI, XML)

- O produto deve suportar especificações avançadas de WS-® (incluindo WS-Security, WS-Coordination, WS-ReliableMessaging, etc).
- O fornecedor deve implementar o conjunto de especificações WS-® através de downloads do kit de ferramentas em até seis meses da data de lançamento das especificações.
- O produto deve dar suporte à assistência personalizada do Web Services através da exposição de APIs SDK, ou fornecer mecanismos de interceptação.
- O produto deve ter suporte inerente para os Web Services, em vez de ter suporte através de um produto separado que é acrescentado posteriormente.
- O acesso a vários recursos de Web Services (p.ex., segurança, garantias transacionais) deve ser suportado através de APIs programáticas e de meios declarativos.
- O produto deve dar suporte a "ativação de Web Services" fácil de interfaces de serviços comerciais existentes.
- O produto deve operar com todos os aplicativos que expõe serviços através das interfaces padrão de Web Services.
- O produto não deve ter extensões patenteadas que possam ir de encontro aos requisitos de interoperabilidade baseada em Web Services.
- O kit de ferramentas Web Services precisa ter passado por um teste rigoroso de workshop para WS-Interoperability para demonstrar a capacidade de interação operacional com os outros kits de ferramentas de Web Services.
- O fornecedor deve ser um membro ativo da organização WS-I.

- **Classificar os requisitos totais** - Trabalhando com os requisitantes principais do aplicativo, o conjunto total de requisitos é classificado. Em um nível grosseiro, cada requisito é considerado como obrigatório, desejável, de baixa prioridade ou não aplicável. Dentro de cada uma dessas categorias, ponderações de importância são atribuídas para permitir controle detalhado das classificações dos requisitos, de forma semelhante a [4,5]. O resultado dessa etapa é uma coleção de requisitos ponderados armazenados na ferramenta de análise de requisitos do i-MATE.
- **Identificar produtos candidatos** - Essa etapa identifica de três a cinco produtos de integração mais prováveis de serem aplicáveis aos requisitos totais do aplicativo. Em alguns casos o cliente já identificou uma lista, baseado em razões técnicas e comerciais. Em outros, usamos a nossa experiência para trabalhar com o cliente para identificar os candidatos mais prováveis.
- **Avaliação de produtos** - Em workshops com os requisitantes principais e representantes de fornecedores de produtos, avaliamos cada um dos produtos candidatos em relação aos requisitos totais. As pontuações são alocadas em relação a cada ponto de requisito para cada produto e capturadas na ferramenta de análise de requisitos. Isso envolve realizar discussões técnicas aprofundadas e percorrer cenários de aplicativos relevantes para entender de modo preciso como os produtos de integração realmente se comportam. Em alguns casos, os recursos e as capacidades do produto podem levar o processo a iterar e refinar as classificações de requisitos. Após todos os produtos terem sido avaliados, a ferramenta de análise de requisitos calcula automaticamente pontuações resumidas ponderadas baseadas em pontuações e ponderações de requisitos individuais Gráficos de resumo também são criados automaticamente para colaborar com relatórios e apresentações eficientes.
- **Análise de cenário** - Variando as ponderações dos requisitos, a ferramenta de análise de requisitos torna trivial explorar diversos cenários e trocas prováveis. Isso pode ser usado para diferenciar melhor os produtos candidatos ou confirmar a adequação de um produto determinado segundo requisitos variados. O resultado dessa etapa é a recomendação de um ou mais produtos que podem satisfazer os requisitos do aplicativo.
- **Protótipo de prova de tecnologia** - Quando o resultado da avaliação do produto não é 100% conclusiva, um rápido protótipo de prova de tecnologia é desenvolvido. O protótipo normalmente implementa um cenário crítico que irá exercitar e/ou enfatizar os requisitos considerados de prioridade mais alta. Até mesmo protótipos bastante simples são ferramentas poderosas que fornecem evidências concretas e indiscutíveis das capacidades do produto. Em vários projetos i-MATE os resultados dos protótipos forneceram a diferenciação final requerida para finalizar a seleção de produtos. De fato, uma fase de protótipo é sempre recomendada mesmo se um produto sair do processo como um líder evidente. No entanto, quando somente um produto for considerado, a tarefa de formação de protótipo não é competitiva e pode ser estruturada mais para validar requisitos principais do aplicativo.

Em termos de recursos, os estágios de Avaliação de Produtos e Protótipo de Prova de Tecnologia invariavelmente consomem a maior parte dos esforços no I-MATE. A avaliação de produto leva em média entre um e três dias por produto, dependendo da familiaridade da equipe do i-MATE com o produto específico. A formação de protótipo é mais variável e depende da complexidade do protótipo desejado. Na maioria dos casos um sistema simples é suficiente e o estágio de protótipos dura menos de uma semana. Em outros aplicativos em que os riscos são mais altos, a formação de protótipos estendeu-se para um mês.

Tabela 1. Analizando Cenários

Custeios	Custos básicos de treinamento/serviços/tecnologia
Mensagem XML e gerenciamento de serviço	Instalações disponíveis para definição do formato da mensagem, definições de interface/contrato de serviço, gerenciamento de serviço
Arquitetura de integração	Recursos arquiteturais centrais, flexibilidade, serviços de criação de eventos, como os serviços são descobertos/integrados/invocados
Adaptadores	Gama e qualidade dos adaptadores disponíveis para integração com sistemas externos
Transformação de dados e Conversão XML	Quão capacitadas são as ferramentas de transformação XML incorporadas? Características de desempenho de processamento, capacidades de mapeamento de dados
Qualidade de Entrega	Fundamentos do Service Bus, modo de operação, qualidade do serviço (p.ex., serviço confiável de mensagens), capacidades de roteamento
Suporte a Web Services	Está em conformidade com os padrões de Web Services? Quão abrangente é o suporte a Web Services? O suporte aos Web Services é inerente ou é acrescentado posteriormente?
Desenvolvimento e Suporte	Como são desenvolvidos e depurados os aplicativos?
Desempenho	Desempenho bruto e questões de escalonamento
Segurança	Autenticação, autorização, criptografia, instalações únicas de inscrição
Serviços de Transação	Instalações disponíveis para suporte de comportamento transacional
Fluxo de Trabalho	Gerenciamento do processo comercial e recursos de automação, monitoramento da atividade comercial
Gerenciamento do sistema	Como são implantados e gerenciados os aplicativos e qual o controle das versões
Técnico	Vários requisitos técnicos.

A base de conhecimento do i-MATE contém um extenso conjunto de requisitos genéricos para tecnologias de middleware mais genéricas, além dos específicos de tecnologias que oferecem suporte a orientação para serviços. Esses requisitos genéricos derivam das experiências práticas do projeto Middleware Technology Evaluation da CSIRO [6], trabalhando com fornecedores de produtos e em consultoria para clientes, como em [7].

Da mesma forma como existem diferentes classes de produtos de middleware, existe uma instanciação diferente da base de conhecimento total para cada classe de produtos. Por exemplo, a base de conhecimento tem versões para tecnologias de integração baseadas em serviços, EAI (Enterprise Application Integration), tecnologias Application Server e tecnologias CORBA. Iremos observar a base de conhecimento da integração baseada em serviços no exemplo a seguir.

Uma análise detalhada do conjunto de requisitos genéricos resultou em cada base de conhecimento ser estruturada como um conjunto de categorias de alto nível que encapsulam vários itens de requisitos individuais. A apresentação da base de conhecimento toda está fora do escopo deste ensaio, mas como exemplo, para a versão de tecnologia de integração baseada em serviços, as categorias poderão aparecer da seguinte forma:

- **Categorias de avaliação de alto nível** - Cada categoria de alto nível contém normalmente entre 10 e 20 requisitos individuais que se relacionam a essa categoria. Por exemplo, a categoria Suporte a Web Services contém requisitos de tecnologia individuais.
- **Suporte a Web Services** - Esses pontos de requisito cobrem recursos detalhados de baixo nível das tecnologias de integração. Todas as soluções de integração baseadas em serviço irão inevitavelmente requerer algumas ou todas essas capacidades. Durante um projeto an i-MATE, o cliente é conduzido pelo conteúdo da base de conhecimento e a importância de cada requisito para o aplicativo do cliente é determinado. Em alguns projetos o cliente é conhecedor da tecnologia e o processo é rápido e direto, demorando menos que um dia. Em outros projetos o cliente depende da equipe do i-MATE para explicar as implicações de muitos dos requisitos e sua importância relativa é definida de forma colaborativa.

Além dos requisitos categorizados, a base de conhecimento do i-MATE é preenchida com avaliações de diversas versões dos principais produtos de middleware. Cada produto da base de conhecimento é classificado em uma escala de 1 a 5 em relação a requisitos individuais. As classificações ocorrem e são mantidas atualizadas por meio de dois mecanismos, como explicado mais adiante e representado na Figura 4.

O primeiro é o projeto MTE, que avalia com rigor tecnologias de middleware usando uma abordagem definida e repetível [6]. Os resultados das avaliações do MTE abastecem diretamente as avaliações da base de conhecimento do i-MATE. O segundo mecanismo são os próprios projetos i-MATE. Os clientes solicitam com freqüência que uma tecnologia de integração ou outro produto ou versão de middleware que não tinha sido avaliado anteriormente seja avaliado durante um projeto de aquisição. Nessas circunstâncias a equipe do i-MATE trabalha com o fornecedor do produto para classificar os recursos do produto. A avaliação resultante estende a cobertura dos produtos na base de conhecimento e eles podem ser reutilizados em projetos subseqüentes.

Ao reutilizar os requisitos genéricos no i-MATE, as organizações economizam o custo de desenvolver o seu próprio conjunto de requisitos de tecnologia de integração. O esforço pode assim ser concentrado em capturar seus requisitos específicos de aplicativo e planejar e projetar a arquitetura orientada a serviços no nível da empresa. Isso economiza tempo e trabalho e ajuda a produzir um resultado de baixo risco.

Análise de troca

Uma ferramenta de análise de requisitos personalizada foi construída para oferecer suporte à análise de troca como parte do processo i-MATE. A funcionalidade básica da ferramenta oferece o seguinte:

- Captura de pontos de requisitos individuais genéricos e específicos do aplicativo, estruturados em categorias de alto nível.
- Captura de classificações de produtos e ponderações de requisitos.
- Cálculo instantâneo de médias ponderadas de categorias de requisitos.
- Relatórios e cálculos instantâneos dos resultados da avaliação com o uso de quadros e gráficos.

Uma imagem de tela da ferramenta de análise de troca é mostrada na Figura 5 . É baseada em um programa de planilha eletrônica. A força principal dessa abordagem é demonstrada durante as fases Avaliação de Produtos e Análise de Cenário Provável do i-MATE. Como a planilha é "viva", qualquer mudança feita nas classificações de uma categoria ou na ponderação de um item de requisito reflete imediatamente nos gráficos que representam as pontuações da avaliação.

Por exemplo, na Figura 6 é mostrada a tela para definir ponderações de categorias de requisitos. Nesse projeto, as categorias Mecanismo de Regras, Desenvolvimento e Suporte e Gerenciamento do Sistema são consideradas como de prioridade mais alta. Essas definições geram um conjunto de gráficos que representam as classificações dos produtos após a avaliação dos produtos estar concluída. Nesse estágio geralmente é conveniente explorar como o resultado da avaliação geral pode variar se uma dessas categorias de alta prioridade for reduzida a um nível de prioridade médio. A alteração de qualquer dos valores de prioridade faz com que a planilha reflita instantaneamente essas prioridades alteradas nos resultados da avaliação. Isso torna viável explorar alternativas rapidamente e confirmar os resultados da avaliação em diversos cenários alternativos.

Conclusão

O processo i-MATE é descrito como um processo para facilitar a avaliação de tecnologias de integração dentro do contexto de implementar uma arquitetura orientada a serviços.

As tecnologias de integração são coleções diversas, complexas e altamente técnicas de produtos que normalmente operam em ambientes comerciais de missão crítica. Também é um investimento de TI significante para garantir integração futura suave. A contribuição principal do i-MATE para facilitar o processo de avaliação da tecnologia da integração está na combinação do seguinte:

- Um conjunto pré-fabricado e reutilizável de requisitos genéricos, baseados na análise das características de componentes middleware.

- Um processo de incorporar requisitos específicos do aplicativo, ponderando os requisitos individuais.
- Suporte de ferramenta para capturar e explorar rapidamente trocas de requisitos e gerar relatórios mostrando como os produtos de middleware se comparam em relação aos requisitos.

A abordagem de integração baseada em serviços contém a chave para a integração uniforme e interoperabilidade no futuro. Se as coisas forem feitas corretamente, não deveremos mais enfrentar os tradicionais problemas de integração de aplicativos corporativos. Com o advento de Web Services e todo o mercado contribuindo e participando do esforço de padronização pela primeira vez no nosso mercado de TI, os Web Services e a arquitetura orientada a serviços contêm a promessa de resolver o desafio da integração de aplicativos corporativos. A integração baseada em serviços é um padrão importante para a implementação dessa visão. A seleção cuidadosa de uma tecnologia de integração para essa finalidade é absolutamente crucial para contribuir para o êxito desse empenho da engenharia de software.

Referências

1. J. Cooper e M. Fisher, Software Acquisition Capability Maturity Model (SA-CMM), Version 1.03, CMU/SEI-2002-TR-010, March 2002, em *http://www.sei.cmu.edu/publications/documents/02.reports/02tr010.html*

2. S. Comella-Dorda, J. C. Dean, E. Morris, P. Oberndorf, A Process for COTS Software Product Evaluation, in Proceedings of 1st International Conference on COTS-Based SystemsICCBSS 2002 Orlando, Florida, pp.86-96, February 4-6, 2002.

3. J. Kontio, A Case Study in Applying a Systematic Method for COTS Selection, in Proceedings of the 18th International Conference on Software Engineering, pp 201-209, IEEE, Berlin, March 1996.

4. J. Kontio, A Case Study in Applying a Systematic Method for COTS Selection, in Proceedings of the 18th International Conference on Software Engineering, pp 201-209, IEEE, Berlin, March 1996.

5. Patricia K. Lawlis et al, A Formal Process for Evaluating COTS Software Products, Computer, Vol. 34, No. 5, May 2001.

6. I.Gorton, A.Liu, Software Component Quality Assessment in Practice: Successes and Practical Impediments, in Proceedings of the International Conference on Software Engineering, Orlando, May 2002, IEEE, pages 555-559.

7. Ian Gorton, Anna Liu, Streamlining the Acquisition Process for Large-Scale COTS Middleware Components, in Proceedings of 1st International Conference on COTS-based Software Systems, Florida, Volume 2255, pp 122-131, Lecture Notes in Computer Science, Springer-Verlag, Feb 2002.

Sobre os autores

Anna Liu é arquiteta no *Microsoft Australia Developer and Platform Evangelism Group*. É especialista em projetos de integração de aplicativos corporativos, apaixonada pela codificação de práticas recomendáveis de engenharia de software e estimuladora da adoção dessas práticas e aprendizado pelas empresas. Anna foi anteriormente uma cientista pesquisadora trabalhando em tempo integral na CSIRO e cientista visitante no Software Engineering Institute, CMU. Tem Ph.D em ciência da computação.

Ian Gorton é pesquisador senior na *National ICT Australia*. Até março de 2004 foi arquiteto-chefe do Departamento de Engenharia e Ciências da Informação no Pacific Northwest National Laboratory do Departamento de Energia dos EUA. Anteriormente trabalhou na Microsoft e IBM e ocupou outros cargos de pesquisas, incluindo a CSIRO. Seus interesses incluem arquiteturas de software, particularmente de sistemas de informação de grande escala e alto desempenho que utilizam tecnologias de middleware COTS (Commercial Off-The-Shelf). Recebeu Ph.D em Ciência da Computação na Sheffield Hallam University.

Aviões, Trens e Automóveis

Simon Guest

Introdução

Hoje existem muitas implementações de Web Services através de várias plataformas e ambientes. A maioria delas compartilha uma coisa em comum: todas usam HTTP como o transporte de base. A natureza onipresente do HTTP ajudou os Web Services a alcançar seu nível de adoção atual. Apesar disso, o HTTP é adequado para todo problema? Existem arquiteturas de aplicativo que se beneficiariam do uso de outros transportes? Quais são as vantagens e desvantagens de se fazer isso? Este artigo se dispõe a responder essas questões e mais, analisando os cenários em que transportes alternativos para Web Services podem oferecer uma solução melhor do que o HTTP.

Como muita gente sabe, o HTTP teve uma longa história antes dos Web Services. Ele é o transporte padrão para navegar páginas da Web desde que as primeiras versões no NCSA Mosaic foram lançadas para o mundo. O HTTP como está é bastante adequado para os Web Services. Como é penetrante, normalmente trabalha bem através de firewalls e servidores proxy, elementos (como WSDL) são fáceis de testar usando HTTP e existem muitas pilhas HTTP e servidores disponíveis nos quais construir implementações. Se também voltarmos e olharmos algumas das "metas de design" iniciais dos Web Services, vemos muito ímpeto em torno de enfrentar publicamente os Web Services. Como resultado, o HTTP é uma escolha perfeita.

Apesar da penetração do HTTP, existem cenários em que ele nem sempre se encaixa. Na minha experiência, eles se dividem em três categorias: *Conexões assíncronas; Aplicativos locais e off-line; Computação ponto a ponto.*

Vamos pegar uma dessas áreas por vez e examinar cenários em que o HTTP pode não ser a correspondência perfeita:

Conexões assíncronas

Contoso Consulting, uma organização fictícia, emprega cerca de 5.000 consultores no mundo inteiro. Em anos recentes, passaram por uma expansão fenomenal, empregando muitos membros de equipes no final da era pontocom. Devido a esse aumento na força de trabalho, um dos problemas que defrontam hoje é o do envio de folhas de ponto. Toda semana cada consultor deve enviar uma folha de ponto para que as contas dos clientes sejam precisas e pontuais.

O modelo atual envolve enviar uma folha de ponto por e-mail (criada usando um modelo em Microsoft Excel) ao departamento de contabilidade. Após a folha de ponto chegar, um membro do grupo de contabilidade o insere no sistema de contabilidade para registrar as horas a serem cobradas. O sistema de contabilidade atualmente é baseado em mainframe.

Como se pode imaginar, esse modelo de enviar folhas de ponto não está bem dimensionado em relação à expansão atual. Apesar da contratação de mais pessoas para o departamento de contabilidade, o método de lidar com as folhas de ponto que chegam por e-mail está se tornando laborioso devido ao processo manual de transportar os dados do Excel para o sistema de contabilidade.

Para ajudar com isso, a TI interna criou um novo serviço de envio de folhas de ponto Usando Web Services, esse serviço fica na frente do mainframe, aceita uma folha de ponto de um consultor e insere automaticamente os detalhes no sistema de contabilidade. O design do Web Service foi mantido simples e um aplicativo Smart Client foi desenvolvido para o envio, como mostrado na Figura 1.

Uma das primeiras metas de design, porém, foi garantir que o envio das folhas de ponto seja realizado de forma assíncrona. O sistema de contabilidade está se tornando antiquado e o pensamento de 5000 consultores enviando suas folhas de ponto no último dia da semana é algo assustador. Para superar isso, o arquiteto do sistema decidiu implementar uma fila de mensagens entre o Web Service e o sistema de contabilidade, com mostrado na Figura 2.

Figura 1. Uma fachada de Web Services é usada para expor o Web Service

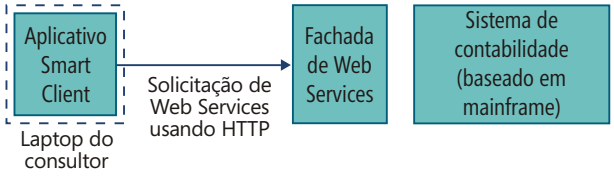


Figura 2. Uma fila de mensagens é usada para agrupar solicitações de clientes ao mainframe

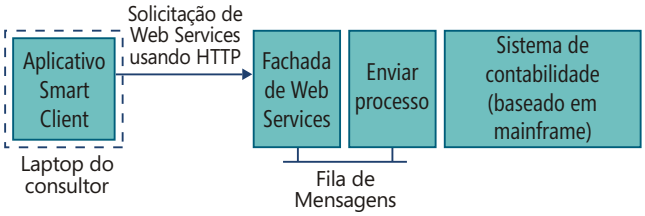


Figura 3. Resposta assíncrona ao cliente é difícil

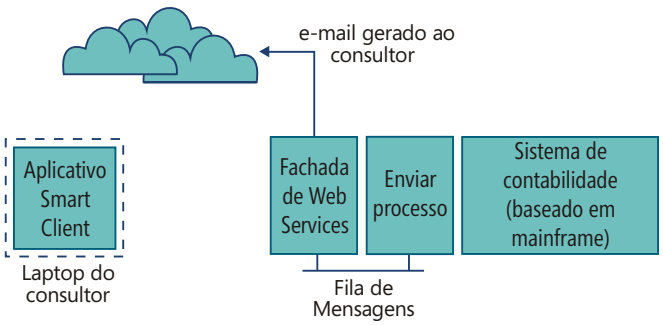


Figura 4. Uma folha de ponto é enviada diretamente à fila de mensagens

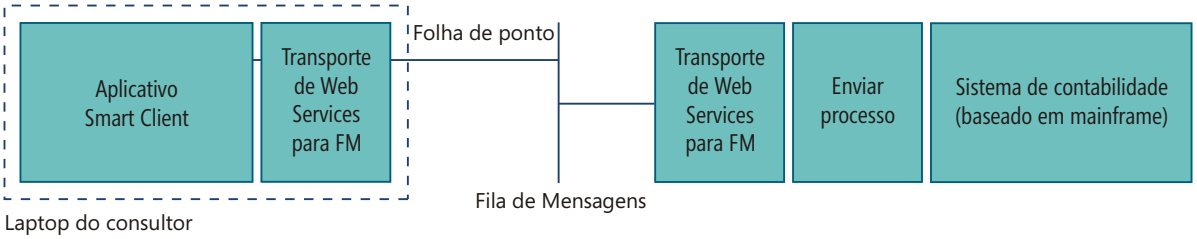
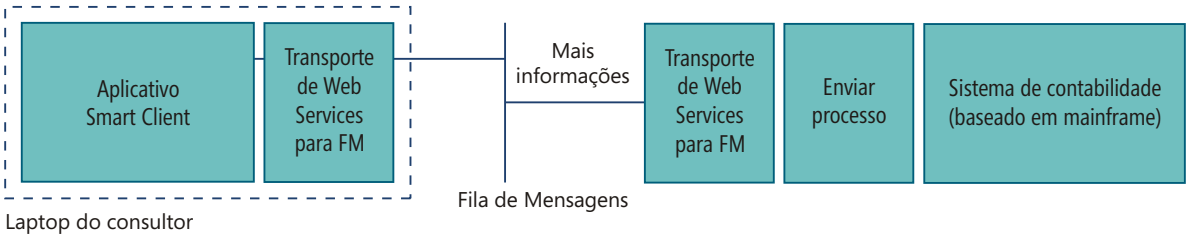


Figura 5. O sistema de contabilidade solicita mais informações usando a fila de mensagens



A responsabilidade da fila será agrupar as solicitações dos clientes antes de enviá-las ao sistema de contabilidade. Esse design assíncrono evitará sobrecarregar o sistema de contabilidade com muitas solicitações simultâneas e, ao mesmo tempo, "liberar" o aplicativo Smart Client para realizar mais tarefas (ou seja, o Smart Client não precisa aguardar até o sistema de contabilidade processar a folha de ponto antes de passar para outras tarefas).

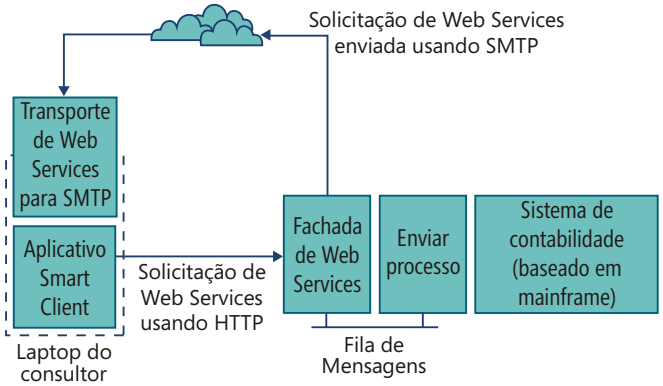
Isso tudo parece bom em princípio, mas o grupo de TI observa um problema potencial. O que acontece se houver um problema com o envio da folha de ponto?

Imagine a seguinte seqüência: Um consultor envia uma folha de ponto usando o Web Service na Figura 2. Tudo funciona e a solicitação da folha de ponto é colocada na fila de mensagens. Após um par de horas (é sexta-feira à tarde e o sistema está ocupado), o sistema de contabilidade lê a folha de ponto na fila de mensagens. Nesse processo descobre-se que o consultor assinalou incorretamente algumas horas em um projeto que anteriormente estava marcado como concluído. O sistema de contabilidade precisa dessa informação antes de poder continuar a processar a solicitação.

Usando o design atual, quais são as opções? O sistema de contabilidade poderia enviar um alerta para um membro da equipe de contabilidade (talvez uma mensagem de sistema) para indicar que necessita de mais informações. O membro da equipe de contabilidade poderia ir atrás do consultor para obter os dados corretos. Isso poderia fechar o círculo, mas ainda é uma tarefa manual; e só irá crescer com o número de pessoas na organização.

Como alternativa, o sistema de contabilidade poderia enviar um alerta diretamente para o consultor, talvez enviando um e-mail ao consultor solicitando as informações adicionais. Novamente, isso deveria fechar o círculo, mas a solicitação de informações ainda está desconectada do processo original. Como o consultor relaciona o e-mail à folha de ponto enviada? Como o e-mail descreve com precisão as informações que estão faltando? Como o sistema de contabilidade correlaciona o envio de uma folha de ponto nova ou modificada com a folha de ponto antiga e o e-mail que foi enviado? Como o sistema de contabilidade prossegue se o consultor simplesmente excluir o e-mail? O que acontece com a folha existente? Existe a possibilidade de esse círculo nunca fechar.

Figura 6. Usando SMTP para a solicitação de Web Services



Vamos voltar uma etapa e perguntar por que foi necessário um e-mail. Por que o sistema de contabilidade não poderia comunicar a informação diretamente ao aplicativo Smart Client? A resposta: HTTP é um protocolo de a solicitação/resposta.

Após a folha de ponto ter sido enviada e a solicitação/resposta HTTP ter sido concluído, como mostrado na Figura 3, é praticamente impossível comunicar de volta com o cliente para obter mais informações. Mesmo se o cliente estiver executando um Web Service local (para aceitar solicitações de Web Services que entram), o que acontece se eles ficarem off-line ou estiverem a atrás de um firewall no site de um cliente? E se o endereço IP e/ou nome de host foi alterado desde a última comunicação? Mais assustador ainda é questão de quem gerencia as implementações do servidor da Web em cada um dos laptops dos 5.000 consultores.

Como o HTTP é um protocolo de solicitação/resposta, e desse modo é muito difícil para o sistema fazer a continuação com o cliente, medidas assíncronas alternativas (ou seja, o e-mail) foram tomadas. Infelizmente, como esse e-mail é efetivamente "desconectado" da solicitação original, muitas vezes é necessário muito mais trabalho para correlacionar o que deve acontecer.

Para ver como podemos criar uma solução usando transportes diferentes do HTTP, vamos analisar um par de abordagens alternativas:

Web Services usando uma fila de mensagens - Chegar à conclusão de que o HTTP era o culpado foi fácil: A nossa primeira abordagem em busca de uma solução requer pensar em um transporte alternativo para substituir ou aperfeiçoar as conexões HTTP do nosso design.

Tomando o design existente, vamos substituir a comunicação HTTP por uma fila de mensagens. A implementação não é importante nesse estágio (poderia ser MSMQ, IBM MQ Series, Tuxedo etc.) desde que seja capaz de tratar as solicitações assíncronas de forma confiável.

É importante observar que ainda estaríamos usando Web Services, estamos enviando e recebendo mensagens SOAP, exceto que elas estão sendo enviadas usando uma fila de mensagens assíncronas em oposição ao HTTP. Como ilustrado, podemos usar algum tipo de transporte ativado por Web Service para a fila de mensagens. Desse modo, como essa arquitetura funciona com o nosso novo cenário?

O Smart Client envia uma folha de ponto (e uma solicitação de Web Service correspondente é criada). Esse pedido é colocado diretamente na fila de mensagens em vez de enviado por HTTP. Após a mensagem ser colocada na fila, o cliente pode desconectar com segurança. (Figura 4)

O Web Service baseado no servidor, por sua vez, irá pegar essa mensagem e comunicar com o sistema de contabilidade para processar a solicitação. No caso em que o sistema de contabilidade precisar solicitar informações adicionais, uma nova mensagem (uma solicitação ao consultor) é colocado na fila; o destinatário é o aplicativo Smart Client. (Figura 5)

Essa mensagem permanecerá na fila até o Smart Client reconectar. Para garantir que a mensagem seja apanhada no momento certo, podemos considerar um serviço em segundo plano no cliente que conecta à fila de mensagens e envia ao consultor a caixa de diálogo "informações faltando" do aplicativo da folha de ponto.

Isso oferece uma solução ao nosso problema de círculo fechado e poderia muito bem oferecer uma abordagem mais automatizada, mas possui uma falha. O Smart Client deve ser capaz de conectar à fila de mensagens para processar os pedidos que entram dos sistemas de contabilidade. A maioria dos fornecedores de filas de mensagens faz isso acessando alguma API de fila de mensagens patenteado. E se o consultor estiver em viagem? E se o consultor tiver acesso à Web e aos e-mails somente em um aeroporto? Essas mensagens não serão apanhadas até ele conectar-se à rede corporativa, e isso poderia ser inaceitável.

Figura 7. Uma fila de mensagens local é usada para manter a solicitação off-line

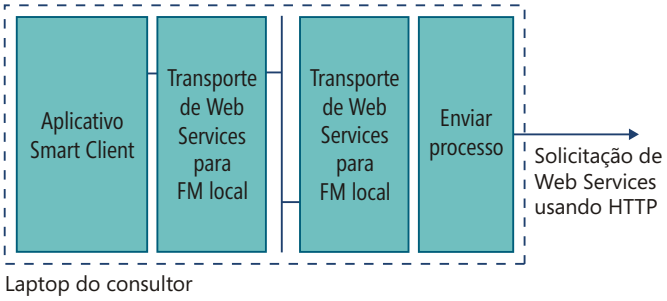


Figura 8. Dois aplicativos da mesma máquina comunicando com HTTP

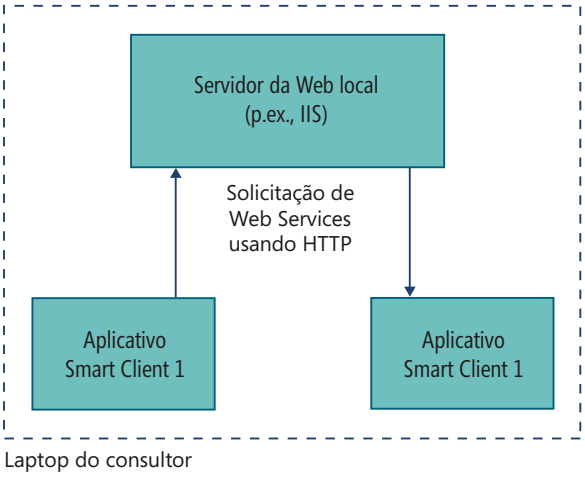


Figura 9. TCP ou In Process fornece uma maneira mais direta de aplicativos locais se comunicarem

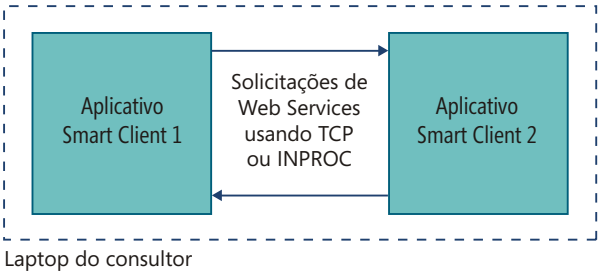


Figura 10. Transporte de Web Services é usado para registrar mensagem para SQL

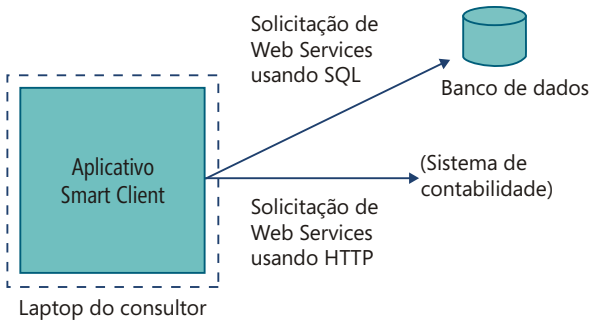
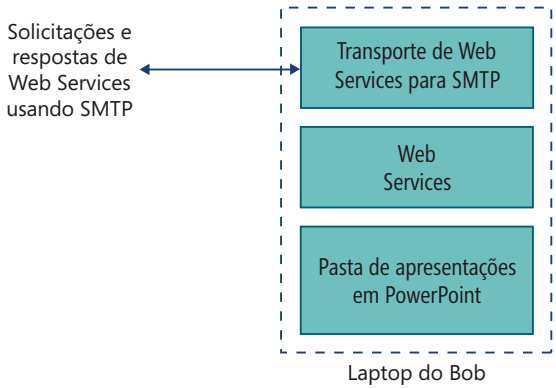


Figura 11. Web Service do Bob processa solicitações e respostas por SMTP



Usando outro transporte, vamos observar uma segunda abordagem:

Web Services usando transportes HTTP e SMTP - Um dos principais problemas com o design original é que o sistema de contabilidade, ao enviar o e-mail ao consultor solicitando mais informações, criou efetivamente um círculo aberto. Esse design depende de o consultor precisar associar manualmente a mensagem de texto que recebe com o processo no aplicativo.

O transporte em si, porém, é razoavelmente eficiente. Com a confiabilidade do e-mail nos dias de hoje, é mais que provável que o consultor terá recebido o e-mail.

Assim, poderíamos considerar um novo design construído sobre isso:

Aqui, uma solicitação de Web Service por HTTP ainda é usada para fazer o envio da folha de ponto. Do mesmo modo que o design original, isso está comprometido com uma fila de mensagens para liberar a conexão do Smart Client. Nesse design, se houver um problema com a folha de ponto, um e-mail é enviado, mas não ao consultor. Em vez disso, um e-mail é gerado contendo uma solicitação de Web Service ao aplicativo Smart Client originador. O que estamos fazendo é iniciar uma solicitação de Web Service para obter as informações adicionais usando SMTP como o transporte.

O Smart Client precisa de um par de modificações para fazer esse trabalho. Precisa haver uma maneira de recuperar a solicitação SOAP usando e-mail; poderia ser um filtro na caixa de entrada do consultor ou uma conta de e-mail separada do aplicativo Smart Client. Segundo, após o e-mail ser recebido, o aplicativo Smart Client deve processá-lo e fazer com que a ação correta ocorra no cliente (uma caixa de diálogo solicitando ao consultor as informações que faltam, por exemplo). Uma vantagem dessa abordagem é que ela usa transportes existentes, permite que o aplicativo de contabilidade inicie uma solicitação ao aplicativo Smart Client e (desde que seja possível acessar e-mail em uma localidade remota) não restringe o consultor a precisar conectar à rede corporativa para enviar relatórios de despesas. Lembre-se também que, como a solicitação é um Web Service, outros padrões (como WS-Security) podem ser igualmente aplicados, fornecendo integridade e sigilo à mensagem embora esteja sendo enviada através de servidores SMTP públicos.

Este conceito de conexões assíncronas também pode aplicar-se igualmente ao cliente. Tomando o nosso exemplo anterior, vamos imaginar que o consultor está pronto para enviar uma folha de ponto. Ele gera a folha de ponto no aplicativo Smart Client e a envia (usando HTTP) ao Web Service.

Isso funciona perfeitamente bem, desde que haja uma conexão com o Web Service. O que acontece quando o consultor envia a folha de ponto de uma localidade onde não há conectividade (ou se estiver no avião a 10.000 m de altura entre um cliente e outro, por exemplo)? Nesse caso, teríamos de pensar em algum tipo de abordagem off-line. Ao ser pressionado o botão "Enviar", o design do Smart Client teria de detectar que não há conexão de rede e a operação seria suspensa ou salva em uma fila ou banco de dados local (Figura 7).

Uma outra alternativa é considerar um segundo transporte. Em vez de usar HTTP diretamente do Smart Client, poderíamos considerar um transporte de enfileiramento local para fornecer essa funcionalidade off-line automaticamente.

Figura 12. Laptop do Joe envia 50 solicitações de Web Services usando SMTP

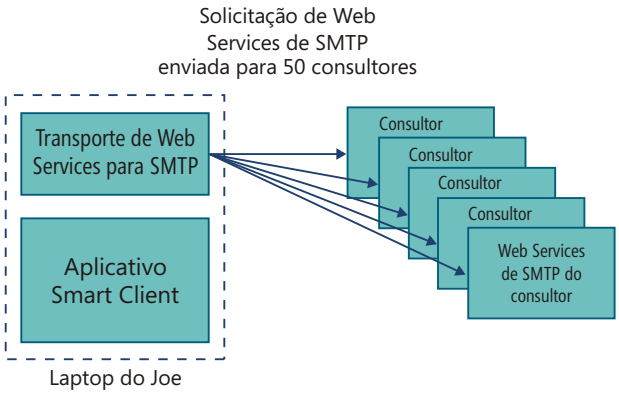


Figura 13. O aplicativo Smart Client processa as respostas recebidas

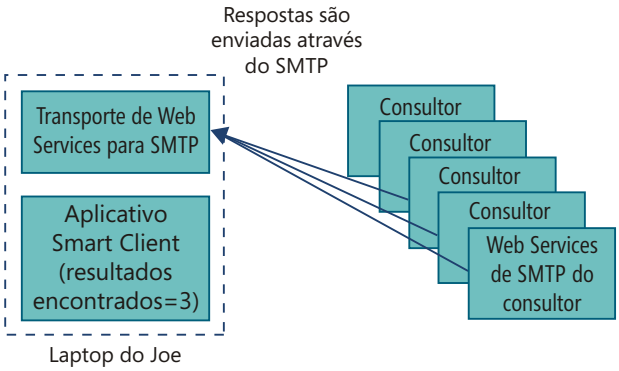


Tabela 1. Transportes alternativos

Problema ao usar HTTP	Transporte alternativo	Vantagens	Desvantagens
Difícil se comunicar novamente com o cliente depois de a solicitação ter sido feita.	Fila de Mensagens (p.ex., MSMQ ou Séries MQ da IBM)	Conexão assíncrona verdadeira	Difícil acessar em um local remoto (requer visibilidade da fila)
Difícil se comunicar novamente com o cliente depois de a solicitação ter sido feita.	SMTP	Conexão assíncrona verdadeira	Requer filtro ou caixa postal alternativa para processar mensagens que chegam
Necessidade de lidar corretamente com o estado de conexão se o serviço HTTP estiver indisponível.	Fila de mensagens local	Enviar mensagens mesmo quando estiver offline	Requer instalação local de fila de mensagens, mais o processo de monitoramento
Solicitações de log para Web Services exigem um código adicional	SQL	Enviar mensagens diretamente para o servidor SQL como transporte alternativo	Precisa de código no Servidor SQL para mapear as solicitações de Web Services para o esquema no Banco de Dados
Web Server requerido para dois aplicativos na mesma máquina	TCP ou em processo	Comunicação direta sem servidor adicional	Requer gerenciamento de soquetes abertos na máquina local (p.ex., pool, firewall local, etc.)
Difícil apresentar Web Services de ponto a ponto entre as organizações (a menos que se abram brechas no Firewall)	SMTP	Pouca infra-estrutura adicional requerida (supondo que o servidor de e-mail já exista)	Segurança difícil de controlar. Bom somente para cenários assíncronos, potencialmente de longo prazo.

Aqui, uma fila local (usando MSMQ, por exemplo) está instalada no laptop do consultor. Em vez de usar HTTP, a solicitação SOAP é colocada na fila por padrão. Um segundo processo, executando potencialmente em segundo plano na máquina do consultor, iria monitorar a instância MSMQ local quanto a novas mensagens e, de forma freqüente, verificaria a possibilidade de ser estabelecida uma conexão com o Web Service com base em HTTP. Quando os dois puderem ser conectados, a mensagem é encaminhada entre os transportes.

Nos aplicativos Smart Client, o uso de transportes de Web Services alternativos também abre outras opções: Imagine que existem dois aplicativos Smart Client executando na mesma máquina que precisam se comunicar (Figura 8). Iniciar as chamadas usando Web Services por HTTP seria exagero, pois iria requerer uma instância local de um servidor da Web e cada solicitação provavelmente iria atravessar a pilha da rede na máquina.

Uma maneira mais eficiente para isso poderia ser usar um transporte baseado em TCP (soquete) ou um In Process (ou transporte de memória compartilhada). Aqui, como mostrado na Figura 9, os dois aplicativos na mesma máquina podem comunicar-se usando solicitações e respostas de Web Service padrão, mas usando um transporte leve e gerenciável.

Além disso, usando o nosso exemplo da folha de ponto, e se quiséssemos implementar uma maneira de registrar todas as solicitações de Web Services (para fins de auditoria)? Provavelmente a nossa abordagem seria criar um registro da mensagem antes de ela sair para o serviço, o que envolveria um filtro ou uma classe para levar a mensagem ao banco de dados.

Isso funciona, mas uma abordagem mais fácil (como mostrado na Figura 10) pode ser implementar um transporte de Web Services para fazer isso. Um transporte poderia usar um banco de dados SQL para registrar solicitações de saída, mas para o aplicativo Smart Client parece apenas um outro transporte.

Aqui, a solicitação de Web Service é enviada usando dois transportes. O primeiro vai para o destinatário pretendido (usando HTTP). O segundo é enviado ao banco de dados para registro usando um transporte SQL.

Finalmente, uma outra área que apresenta grande potencial para transportes de Web Services alternativos é computação ponto a ponto. Vejamos um exemplo:

Bob é um consultor da Contoso. Em seu laptop ele tem um diretório de slides em PowerPoint que usa para apresentações a clientes. Esse diretório o acompanha onde quer que ele vá. Está sendo aprimorado constantemente e deve trabalhar tanto em cenários on-line quanto off-line.

Sendo um bom cidadão, Bob deseja compartilhar esses slides em PowerPoint com seus colegas, tanto dentro da empresa quanto com membros de outras organizações. Muitas pessoas o procuram por e-mail perguntando se ele possui um slide específico sobre um tópico, e pesquisar e responder a eles consome boa parte do tempo de Bob.

Bob está considerando construir um Web Service centralizados para hospedar seus slides em PowerPoint. Isso deverá estar disponível a todos, mas ele deve ser capaz de acessá-lo em cenários off-line. Ele reconsidera as etapas necessárias para implementar um serviço assim.

1. Configuração de um servidor central. Bob precisará pegar seu diretório de slides e hospedá-lo centralmente em algum lugar. Isso inclui não apenas encontrar espaço em disco suficiente, mas também uma consideração sobre gerenciamento de backups e atualizações com as versões mais recentes.

2. Expondo um Web Service. Com a configuração do servidor, Bob vai precisar instalar um servidor da Web na máquina, criar um Web Service e trabalhar com o grupo de TI local para garantir que está hospedado corretamente por trás do firewall da Contoso (provavelmente na DMZ).

3. Criar um aplicativo Smart Client para acessar. Bob está pensando em criar um aplicativo Smart Client que irá permitir que ele mantenha uma versão off-line dos slides que necessita a qualquer momento.

Bob pensa nisso, parece um trabalho imenso, além do que não tem certeza se poderá ser bem dimensionado. E se os outros 5000 consultores da organização desejarem fazer algo parecido? Precisarão passar pelos mesmos processos? E se eles não possuírem um conhecimento técnico tão bom quanto Bob?

Bob pára um pouco e pensa porque deseja fazer isso; o sistema atual funciona muito bem, a questão é que ele fica inundado com solicitações por e-mail sobre apresentações feitas recentemente.

Ele poderia talvez criar um Web Service no seu laptop para cuidar dessas solicitações que entram; o Web Service poderia pesquisar seu diretório de slides em PowerPoint e recuperar alguns determinados para os clientes. O problema com essa abordagem usando HTTP é que o laptop do Bob precisa estar ligado e acessível para esse trabalho. Geralmente Bob passa muito tempo fora do escritório e como ele permite acesso ao seu laptop através de um firewall para clientes externos? Está parecendo bastante difícil de gerenciar.

Após ler sobre o uso de transportes alternativos de Web Services, Bob cria um novo design:

Ele vai criar um Web Service para o seu laptop, mas em vez de aceitar conexões de entrada por HTTP, irá usar SMTP (e-mail), como mostrado na Figura 11. Os clientes podem enviar-lhe solicitações de Web Services para pesquisar e recuperar seu armazenamento local de arquivos PowerPoint. Para isso, Bob irá criar um pequeno aplicativo Smart Client que gera essas solicitações.

A beleza desse design é que Bob e outros podem agora aproveitar a funcionalidade distribuída que o e-mail oferece. Bob compartilha seu novo aplicativo de Web Service com 50 outros consultores da Contoso (Figura 12). O que temos agora é uma maneira bastante dinâmica de usar Web Services para pesquisar arquivos PowerPoint mantidos localmente em uma série de máquinas.

Por exemplo, Joe está procurando uma apresentação PowerPoint sobre o tema C#. Ele insere a consulta "C#" em um aplicativo Smart Client. Isso cria uma solicitação de Web Service que é enviada usando SMTP para uma lista de distribuição de e-mails que contém os 50 consultores que executam o Web Service do Bob.

Quando a mensagem é recebida, o Web Service que executa em cada um desses laptops realiza uma busca com base nos critérios do Joe. A lista de resultados é enviado de volta ao aplicativo de chamada do Joe, que pode exibi-los à medida que as respostas são recebidas (novamente, usando SMTP).

Joe pode agora começar a pesquisar os resultados à medida que eles retornam (lembre-se, ele não precisa que todos respondam, apenas pessoas suficientes que possuam o slide em PowerPoint que ele procura). Quando ele encontra o slide correto, uma solicitação semelhante, usando Web Services por SMTP, pode ser feita para solicitar realmente a apresentação (Figura 13).

Quando considerar transportes alternativos

As abordagens que você acompanhou neste artigo podem levantar mais perguntas do que as respostas que apresenta. Temos a esperança, porém, de que você possa observar que o uso de Web Services com transportes alternativos pode abrir um novo leque de aplicações que até agora estavam restringidas pelo uso do HTTP.

Uma das dúvidas que você pode ter é "Quando eu deveria implementar um transporte diferente do HTTP?" Para ajudar a responder e para resumir os cenários abordados neste artigo. Embora seja uma área relativamente nova, um progresso significante está sendo feito na implementação de transportes alternativos para Web Services (Tabela 2).

Figura 2. Implementando transportes alternativos

Transporte	Desrcrição
"Indigo"	Indigo é o nome de código da próxima geração de ambiente de computação distribuída da Microsoft. Indigo oferece a promessa de vários transportes para Web Services, junto com um modelo de programação unificado. Indigo oferece suporte de forma nativa a HTTP, TCP e MSMQ no CTP de março de 2005. O modelo de programação permite uma interface fácil de estender para outros transportes.
WSE (Web Services Enhancements)	Para quem deseja implementar isso hoje, transportes alternativos também podem ser realizados usando WSE. WSI fornece uma API chamada de transporte personalizado, que permite que transportes diferentes do HTTP sejam usados. Transportes personalizados hoje incluem amostras de MSMQ (http://www.codeproject.com/cs/webservices/SoapMSMQ.asp), IBM MQ Series (http://workspaces.gotdotnet.com/wsemqs), SMTP (http://hyperthink.net), UDP (http://dynamic-cast.com), TCP, e In Process (as duas amostras acompanham o WSE).* *Observe: Interoperabilidade usando TCP não recebe suporte com o uso de WSE.
API JMS (Java Message System)	JAXMail (parte do Sun JWSDP) é uma extensão JAX-RPC (Java Web Services) para fornecer suporte ao protocolo SMTP.
JAXMail (API Java para XML Mail)	Uma série de fornecedores de servidor de aplicativos Java estão agora fornecendo suporte a Web Services por meio de JMS. Isso permite que solicitações e respostas SOAP sejam processadas em uma fila JMS.

Sobre o autor

Simon Guest é Gerente de Programas da equipe Estratégia de Arquitetura na Microsoft Corporation e é especializado em interoperabilidade e integração. Simon possui Mestrado em Segurança de TI na University of Westminster, Londres, e é autor do Microsoft .NET and J2EE Interoperability Toolkit (Microsoft Press, setembro de 2003).
Simon pode ser contatado em seu blog: *http://www.simonguest.com*.

Habilitando Computação na Escala da Internet

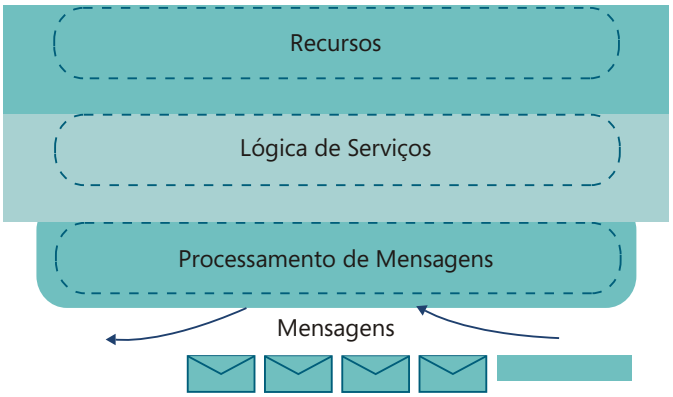
Savas Parastatidis e Jim Webber

Introdução

HPC (High-performance computing, Computação de alto desempenho) evoluiu de uma disciplina preocupada unicamente com a execução eficiente de código em arquiteturas paralelas para alinhar-se mais de perto com a área de sistemas distribuídos. A HPC moderna preocupa-se tanto com o acesso a dados e dispositivos especializados em redes amplas quanto com o processamento de números o mais rápido possível. O foco da HPC mudou para permitir a utilização transparente e mais eficiente possível de uma ampla variedade de recursos tornados disponíveis pelas redes, de maneira tão uniforme quanto a rede elétrica fornece eletricidade. Tal visão requer investimento intelectual e arquitetônico significativo. Neste artigo exploramos uma abordagem orientada a serviços para habilitar aplicativos de alto desempenho em escala de Internet com base no trabalho concluído como parte do programa UK e-Science de £250 milhões.

De redes a estações de trabalho através da Internet [1], a comunidade da computação de alto desempenho há muito defende a composição de recursos de computação individuais em uma tentativa de fornecer qualidade de serviço mais alta (por exemplo, em termos de tempo de processamento, capacidade do armazenamento de dados, largura de banda/latência, acesso remoto a instrumentos, integração de algoritmo especial, etc.). Em anos recentes essa progressão tem sido conduzida pela visão de "Computação em grade", em que o poder de computação, a capacidade de armazenamento e a funcionalidade especialista de dispositivos em rede arbitrários deverão ser tornados disponíveis sob demanda para qualquer outro dispositivo conectado que tenha permissão de acesso a eles.

Figura 1. A estrutura de arquétipo de um serviço



Ao mesmo tempo, a comunidade de sistemas distribuídos tem trabalhado em princípios e tecnologias de design para a integração em escala de Internet (Web e Web Services, por exemplo). Recentemente o termo SOA (Arquitetura Orientada a Serviços) surgiu como uma terminologia popular em algumas partes devido à publicidade que cercou a introdução dos Web Services. Embora os Web Services sejam percebidos como uma tecnologia que permite a construção de aplicativos orientados a serviços, devem ser tratados como uma tecnologia de implementação do conjunto de princípios que constitui orientação a serviços. A promessa de SOA e dos Web Services é permitir acoplamento flexível, robustez, escalabilidade, capacidade de extensão e interoperabilidade. Esses são exatamente os recursos de uma estrutura global de "Computação em grade": uma palavra-chave popular usada para referir-se à computação distribuída de alto desempenho ou computação em escala de Internet.

Nas seções a seguir descreveremos computação em grade e orientação a serviços. Discutiremos em seguida como aplicativos de alto desempenho podem ser projetados, implantados e mantidos com o uso de orientação a mensagens e integração baseada em protocolos. Depois disso apresentaremos a nossa abordagem de como aplicativos HPC de larga escala podem se relacionar com uma grande quantidade de recursos e expressar de uma maneira seja consistente com os princípios de SOA.

Computação em grade

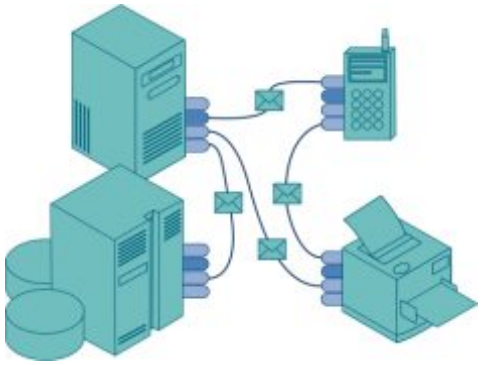
O termo "Computação em grade" é sobrecarregado e possui significados diferentes para comunidades (e fornecedores) diferentes. A seguir, algumas das interpretações mais comuns:

Computação sob demanda; Computação de utilitário; Computação contínua; Interconectividade de supercomputadores; Computador mundial virtual; SETI@home e ClimatePrediction.net; (Projetos estilo BOINC [2]); Organizações virtuais.

Adotamos o ponto de vista de que computação em grade é sinônimo de computação em escala de Internet com foco na exploração dinâmica de recursos distribuídos de computação de alto desempenho. Ao construir aplicativos e infra-estrutura em grade estimulamos a aplicação dos mesmos princípios, técnicas e tecnologias típicos da prática moderna de sistemas distribuídos, com orientação a serviços como o paradigma arquitetônico de opções e Web Services como a tecnologia de implementação.

Embora a orientação a serviços não seja um paradigma arquitetônico novo, o advento de Web Services revigorou o interesse na abordagem. No entanto, é uma concepção incorreta considerar os Web Services uma forma de mágica de software que de algum modo encurrala automaticamente o arquiteto em uma solução de acoplamento flexível que é redimensionável, robusta e confiável. Certamente é possível (e de modo geral altamente desejável) construir aplicativos orientados a serviços com kits de ferramentas e protocolos de Web Services; é igualmente possível construir aplicativos que violem cada um dos princípios de arquitetura e fundamentos da SOA.

Figura 2. Aplicativos em rede são construídos por meio da troca de mensagens entre serviços hospedados em dispositivos. Neste exemplo, um aplicativo que executa em um dispositivo móvel utiliza os recursos distribuídos por meio de serviços que executam em uma estação de trabalho, um banco de dados e uma impressora.



Como os pesquisadores e desenvolvedores deram novos nomes a seus trabalhos para estar em voga com as palavras da moda, o termo Arquitetura Orientada a Serviços (SOA) ficou diluído e impreciso. Devido à falta de uma definição de serviço com aceitação ampla, propomos o seguinte:

Um serviço é a manifestação lógica de algum recurso físico ou lógico (por exemplo, bancos de dados, programas, dispositivos, seres humanos etc.) e/ou alguma lógica de aplicativo que esteja exposta na rede; e os serviços interagem trocando mensagens.

Os serviços consistem em alguns recursos (dados, programas ou dispositivos, por exemplo), lógica de serviço e uma camada de processamento de mensagens que trata das trocas de mensagens (Figura 1). As mensagens chegam ao serviço e são processadas pela lógica de serviço, utilizando os recursos do serviço (se houver) como necessário. As implementações de serviço podem ser em qualquer escala: de um processo de sistema operacional único a processos de negócio no nível da empresa.

Os serviços podem estar hospedados em dispositivos de capacidade arbitrária (por exemplo, estações de trabalho, bancos de dados, impressoras, telefones, assistentes digitais pessoais etc.) fornecendo diferentes tipos de funcionalidade a um aplicativo baseado em rede. Isso estimula o conceito de um mundo conectado em que nem um único dispositivo e/ou serviço está isolado. Aplicativos interessantes são construídos por meio da composição de serviços e a troca de mensagens (Figura 2).

Figura 2. Aplicativos em rede são construídos por meio da troca de mensagens entre serviços hospedados em dispositivos. Neste exemplo, um aplicativo que executa em um dispositivo móvel utiliza os recursos distribuídos por meio de serviços que executam em uma estação de trabalho, um banco de dados e uma impressora.

Mensagens

Mensagem é a unidade de comunicação entre serviços. Os sistemas orientados a serviços não expõem abstrações como classes, objetos, métodos, procedimentos remotos, mas baseiam-se em torno do conceito de transferência de mensagens. Naturalmente, transferências únicas de mensagens possuem uma utilidade limitada, por isso existe uma tendência de uma quantidade de transferências de mensagens serem agrupadas logicamente para formar MEPs (Message Exchange Patterns) (por exemplo, uma mensagem de entrada e uma de saída que estejam relacionadas podem formar um MEP de "solicitação-resposta") para oferecer suporte a interações mais ricas. Os MEPs são agrupados para formar protocolos que capturam o comportamento de envio de mensagens de um serviço (às vezes conhecido como uma conversão) de uma interação especifica . Esses protocolos podem depois ser descritos em contratos e publicados para ajudar a integração com o serviço (por exemplo, em WSDL ou SSDL [3]).

Protocolos e contratos

O comportamento de um serviço em um aplicativo distribuído é capturado por meio do conjunto de protocolos ao qual oferece suporte. A noção de protocolo é um afastamento do mundo orientado a objetos tradicional, em que a semântica comportamental está associada a tipos, exposta por meio de métodos e unida a nós de extremidade específicos (o ponto de acesso de instâncias específicas). Em vez disso, um protocolo descreve o comportamento visível externamente de um serviço somente em termos das mensagens, padrões de troca de mensagens e ordenação dos MEPs que recebem suporte do serviço.

Os protocolos geralmente são descritos por meio de contratos que os serviços seguem. Contrato é uma descrição da diretiva (requisitos de segurança ou recursos de criptografia, por exemplo) e das características da qualidade de serviço (suporte a transações, por exemplo) que um serviço requer e/ou ao qual oferece suporte, além do conjunto de mensagens e MEPs que transportam informações funcionais para o serviço e do serviço.

Da empresa...

Muitas organizações estão percebendo o custo-benefício da utilização de agrupamentos de estações de trabalho como plataformas alternativas para instalações especializadas de supercomputador para suas necessidades de computação de alto desempenho. Até recentemente essas soluções baseadas em agrupamento eram tratadas como recursos de computação e/ou armazenamento dedicados. As empresas estão agora procurando ganhar em termos de custo mais baixo e desempenho usando a capacidade de processamento ociosa, a capacidade de armazenamento distribuído e outros recursos disponíveis de sua infra-estrutura implantada baseada em estação de trabalho, uma abordagem comumente chamada de "computação em grade intracorporativa".

Aqui exploramos agrupamentos dedicados e como a orientação a serviços pode ser usada para construir essas soluções antes de propor uma abordagem para a construção de arquiteturas distribuídas intracorporativas de alto desempenho.

Figura 3. Integrando recursos corporativos para atender os requisitos de alto desempenho dos aplicativos

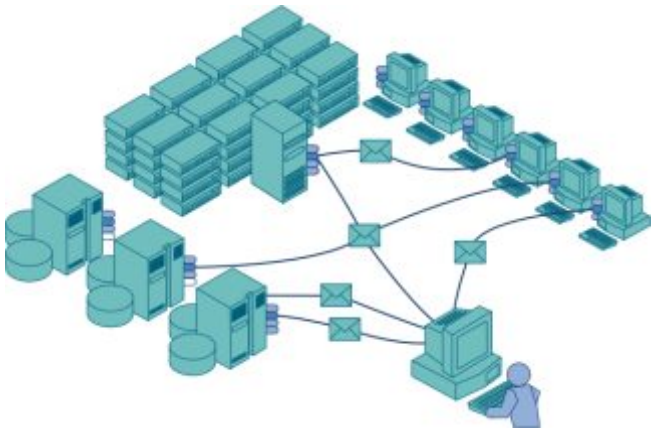
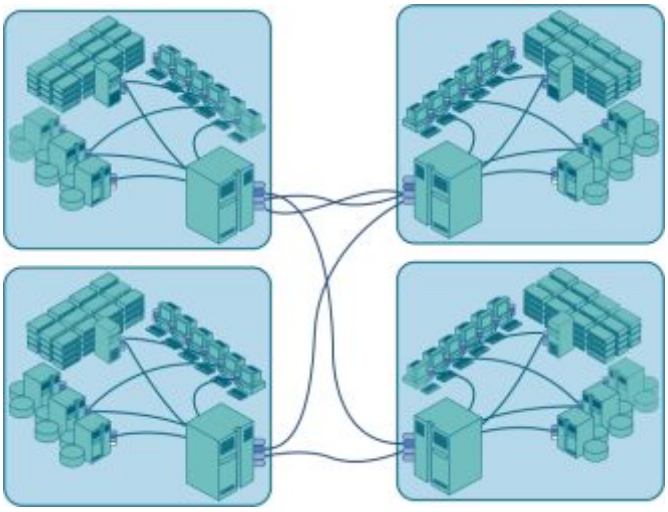


Figura 4. Um exemplo de um aplicativo intracorporativo orientado a serviços com partes diferentes da empresa sendo representadas como serviços



Agrupamentos dedicados

Soluções baseadas em hardware construído com essa finalidade para computação de alto desempenho não são incomuns em um domínio administrativo de organizações com requisitos de computação de alto desempenho. Essas soluções geralmente são implementadas por um ou mais agrupamentos de estações de trabalho com interconexões de alta velocidade (por exemplo, Myrinet, SCI, Gigabit Ethernet etc.). Algumas dessas soluções tentam fornecer uma imagem de computador único aos aplicativos por meio da implementação, em hardware ou software, de técnicas que ocultam a distribuição de CPUs, memória e armazenamento. Uma abstração de programação familiar é apresentada aos desenvolvedores, a de multiprocessamento simétrico com memória compartilhada. No entanto, essas abordagens têm uma tendência de limitar a escalabilidade dos nós de computação, o que pode tornar-se um problema em determinados tipos de aplicativos paralelos. Soluções especializadas de middleware orientadas a mensagens (MPI, por exemplo) geralmente são empregadas para tratar do problema da escalabilidade, mas ao custo de exigir gerenciamento explícito do grau de paralelismo pelo aplicativo. Existem muitos trabalhos na literatura da computação paralela que discutem as vantagens e desvantagens dos paradigmas de memória compartilhada versus transmissão de mensagens para aplicativos paralelos.

Agrupamentos dedicados para computação de alto desempenho geralmente são considerados e gerenciados como recursos únicos. Para permitir uma melhor utilização desses recursos, é preferível uma abordagem baseada em serviços. Por exemplo, o acesso a recursos geralmente é controlado por um serviço de submissão/enfileiramento/planejamento de trabalho que garante ótima utilização do sistema. Tecnologias de Web Services podem ser usadas para a implementação desses serviços e realmente se beneficiam do investimento significativo em ferramentas, suporte eficiente em tempo de execução, documentação e educação do usuário na área de Web Services. A natureza de composição das tecnologias dos Web Services faz com que as necessidades não funcionais da qualidade de serviço da implementação (por exemplo, sistema confiável de mensagens, segurança, transações etc.) sejam incorporadas mais facilmente em um ambiente heterogêneo.

Roubando ciclos de estações de trabalho de equipe

Uma abordagem que ganhou projeção significativa recentemente é a implantação de tecnologias de roubo de ciclos implementadas por middleware especializado, como Condor [4]. Esse middleware permite a distribuição e o gerenciamento de trabalhos de computação em estações de trabalho ociosas, enquanto permite que uma estação de trabalho seja reclamada quase instantaneamente pelo seu usuário com base em console quando ele começa a usar o computador, pois o ladrão de ciclos é suspenso, desligado ou migrado para outra estação de trabalho. No entanto, a maioria das implementações atuais de software de middleware que suportam essas instalações ainda não aproveita os protocolos de qualidade de serviço interoperáveis e de composição. Como resultado, fica difícil criar soluções interoperáveis e uniformes para computação de alto desempenho dentro da empresa.

Na computação de alto desempenho intracorporativa, cada estação de trabalho, sistema de gerenciamento de banco de dados, dispositivo etc. revela alguma funcionalidade como serviço. Construir esse middleware usando os princípios de orientação a serviços pode resultar em implantações que podem crescer para milhares de estações de trabalho. Uma abordagem orientada a serviços pode aumentar a flexibilidade, gerenciabilidade e valor dessas soluções, pois um grande conjunto de unidades de funcionalidade/comportamento amplamente aceitas, tornadas disponíveis como protocolos, pode ser aproveitado (por exemplo, segurança, transações, sistema de mensagens confiável, orquestração etc .) e realmente existe um empenho para fazer exatamente isso com os sistemas existentes (por exemplo, o Condor BirdBath [5]).

Abordagem conceitual de computação HPC intracorporativa baseada em Windows

Um conjunto de estações de trabalho baseadas em Windows usadas pela equipe são parte do domínio do Active Directory "CPUs subutilizadas". A empresa deseja aproveitar as capacidades de computação dessas estações de trabalho durante seu período ocioso (durante a noite, por exemplo). O administrador de domínio promove a implementação. Net de um conjunto de Web Services que fornecem submissão, monitoramento e gerenciamento de trabalhos enviados para estações de trabalho por meio do Active Directory. Esses serviços aproveitam a plataforma middleware de Web Services de base (Indigo, por exemplo) para seus requisitos de recursos de segurança e qualidade de serviço (por exemplo, notificação, serviço de mensagens confiável, transações etc.).

Somente os usuários que pertencem ao domínio "CPUs subutilizadas" têm a permissão de enviar trabalhos. WS-Security é usado para a autenticação e os requisitos de criptografia de mensagens com os registros do Kerberos recuperados do Active Directory. Além das estações de trabalho, existe também um agrupamento dedicado para os requisitos de alto desempenho da empresa e um centro de dados. Os Web Services instalados nesses recursos revelam funcionalidades de computação e armazenamento de dados para a rede. Os aplicativos da empresa são projetados de forma a poderem descobrir e utilizar dinamicamente qualquer recurso de computação distribuído dentro da empresa. Portanto , assim que a funcionalidade é ativada, os aplicativos de computação intensa podem começar automaticamente a aproveitar a infra-estrutura distribuída. Os usuários desses aplicativos não ficam cientes dos recursos realmente usados.

As futuras instalações em grade intracorporativas serão construídas em torno de serviços padrão fornecidos pelos sistemas operacionais de base. Protocolos de aplicativo como WS-Eventing e WS-Management serão implementados e fornecidos como padrão que soluções de tipo grade podem facilmente implementar e implantar.

Um exemplo de abordagem conceitual à computação HPC intracorporativa usando Web Services está descrito na barra lateral e ilustrado na Figura 3.

Identificamos um conjunto incompleto de serviços genéricos, funcionalidades e recursos que pode ser oferecidos e/ou suportados por cada dispositivo da rede (Tabela 1). Naturalmente, as funcionalidades específicas de domínio do aplicativo também terão de receber suporte (por exemplo, um serviço BLAST instalado em um servidor poderoso para realizar análises de bioinformática ou um serviço que implementa um algoritmo de estimativa para uso na indústria do petróleo).

Agrupamentos de agrupamentos na empresa

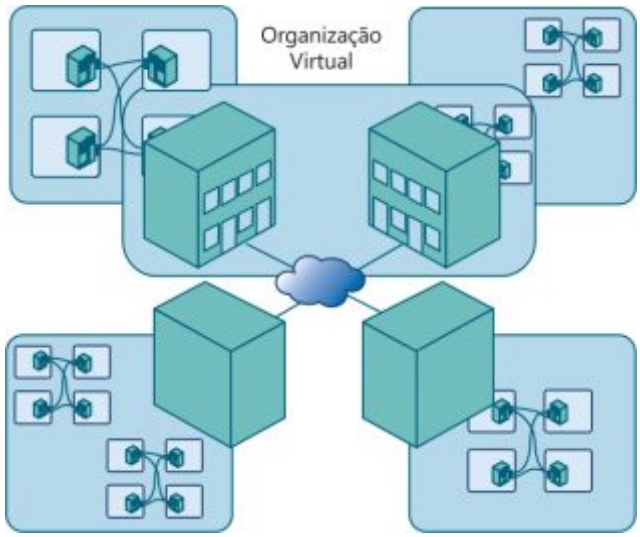
As empresas maiores podem estar interessadas não apenas em implementar somente soluções de agrupamento único ou simplesmente exigir novamente o poder de processamento ocioso de partes da sua infra-estrutura organizacional. Em vez disso, podem desejar concentrar-se no encapsulamento de conjuntos inteiros de recursos de computação que estão por trás de serviços de alto nível que, quando reunidos, podem permitir um nível de integração que anteriormente era difícil e demorado devido à quantidade de tecnologias diferentes implantadas.

A abordagem ao arquitetar soluções intracorporativas de alto desempenho é semelhante a quando o foco é na construção das soluções baseadas em agrupamentos discutidas anteriormente (os problemas de escalabilidade, acoplamento flexível, composição etc. aplicam-se igualmente). Protocolos de qualidade de serviço como segurança (para autenticação, autorização e contabilidade) transações, sistema de mensagens confiável, notificações etc., são todos parte da infra-estrutura de base e podem ser usados sem modificação não importa o tipo de solução implementado. Além disso, o conjunto de serviços usado para soluções intracorporativas também pode ser usado sem modificações (por exemplo, gerenciamento da credencial do usuário, gerenciamento de sistemas, implantação de aplicativos e serviços, suporte a fluxo de trabalho, armazenamento e arquivamento de dados, sistema de mensagens etc.).

Como anteriormente, ainda existe a necessidade de os serviços oferecerem acesso a recursos de dados e de computação, implementarem agendamento, visualização, funcionalidade de algoritmo especializado etc., dependendo do tipo de aplicativo que estiver sendo implementado. Desta vez, porém, os serviços estão em um nível mais alto de abstração porque coleções inteiras de recursos estão encapsuladas (Figura 4).

Observamos que embora a complexidade dos serviços tenha aumentado em relação aos que usávamos ao construir uma solução de agrupamento, a complexidade da arquitetura não aumentou e os princípios e diretrizes permanecem os mesmos. O nosso aplicativo distribuído ainda se liga a mensagens que estão sendo trocadas e nenhuma suposição é feita sobre um entendimento no nível intracorporativo das interfaces e comportamentos dos diversos componentes. Os arquitetos projetam aplicativos por meio da descrição de mensagens e da definição de protocolos que capturam o comportamento de serviços.

Figura 5. Uma ilustração de como as empresas podem ser unidas



Observamos que à medida que a granularidade dos serviços aumenta, a necessidade de aumentar a granularidade das trocas de mensagens é maior. A rede custa caro, por isso os arquitetos precisam projetar seus protocolos e suas mensagens de forma apropriada. À medida que o grau de distribuição de um aplicativo aumenta, a necessidade de acoplamento flexível também aumenta. Em um agrupamento único ou em ambientes corporativos menores, o controle completo da infra-estrutura e do conjunto de tecnologias implementadas é possível, mas em uma solução no nível da empresa (ou maior) é imperativo que a integração ocorra por meio de protocolos padrão.

Também, é claro que à medida que a complexidade da escala de um aplicativo aumenta, a funcionalidade dos seus serviços torna-se mais abstrata ainda. Dos serviços que revelam funcionalidade especifica à rede (execução de processo e gerenciamento de estação de trabalho remoto, por exemplo) ou fornecem o acesso a um recurso (sistema de banco de dados e sistema de arquivos, por exemplo), avançamos para serviços que oferecem suporte à agregação de funcionalidades (filas de trabalhos, filas de mensagens, gerenciamento de agrupamentos, por exemplo) ou agregação de recursos (rede de área de armazenamento e federação de banco de dados, por exemplo).

Como as interações entre os serviços tornam-se mais rude de forma a minimizar o efeito dos custos da comunicação entre as diferentes partes dos aplicativos e os serviços tornam-se mais abstratos e de granularidade mais grossa com relação aos recursos que encapsulam, devemos projetar aplicativos usando blocos de construção maiores. Quanto maior a escala de um aplicativo distribuído, mais importante é desenvolver mecanismos declarativos, baseados em protocolo e de granularidade grossa para descrever comportamento. É nesse estágio que fluxos de trabalho, contratos e diretivas tornam-se ainda mais significativos. Orquestração de serviços e processos de negócios abstratos tornam-se necessários, por isso tecnologias relevantes como WS-BPEL tornam-se uma parte importante do conjunto de ferramentas do arquiteto.

... para a Internet

Da mesma maneira como nenhum dispositivo é uma ilha inacessível dentro de um domínio administrativo, as empresas e organizações não estão isoladas. Como é o caso no nosso mundo físico, as empresas negociam umas com as outras, as organizações interagem e as instituições governamentais colaboram. Serviços e interações são fundamentais para as nossas atividades cotidianas (o serviço bancário, o serviço postal, um serviço de agente de viagens, por exemplo). Nada mais natural, ao modelarmos essas atividades em um mundo computadorizado, seguirmos uma abordagem arquitetônica semelhante à adotada no mundo real.

Como se poderia esperar, no que se refere a aplicativos de escala muito grande com foco no fornecimento de alto desempenho, a natureza do aplicativo pode levar-nos a designs indiferentes com estratégias diferentes em comparação a uma situação intracorporativa. Como é o caso no mundo real , contratos e acordos de nível de serviços são estabelecidos para controlar as interações entre empresas. Organizações virtuais podem ser estabelecidas, da mesma maneira, como alianças são formadas entre empresas no mundo físico, para enfrentar as necessidades de alto desempenho dos aplicativos das entidades participantes. Realmente, um aplicativo distribuído de alto desempenho pode refletir uma aliança no mundo real entre empresas como ilustrado na Figura 5 (uma série de institutos de pesquisas juntando forças para resolver um grande problema científico, por exemplo).

Para as organizações virtuais tornarem-se viáveis, problemas como representações digitais de acordos, negociações de contratos, não rejeição de interações, federação de credenciais de usuário, diretivas e fornecimento combinado de qualidade de serviço, devem ser resolvidos. Descrições de alto nível baseadas em fluxo de trabalho podem ser usadas para representar o comportamento da organização virtual ou para coreografar interações de negócio através da empresa. Os aplicativos nesse espaço, quando corretamente projetados e implementados, podem tornar-se realmente em escala de Internet.

Embora o tipo de serviços encontrados na empresa, como enfileiramento de serviço, agendamento, intermediação de recursos, acesso a dados e integração, visualização etc. ainda possam ser necessários, ao avançarmos para a escala da Internet devemos tomar cuidado quanto à maneira como esses serviços são implementados e implantados. Soluções centralizadas (um único serviço de armazenamento de dados, por exemplo) ou comportamentos de acoplamento rígido (transações de longa duração entre organizações ou exposição direta de estado, por exemplo) devem ser evitados. Naturalmente, pode-se argumentar que Google e Amazon são exemplos espetaculares de repositórios centralizados. Isso é verdade, mas também é o caso que esses são repositórios lógicos que já usam soluções redimensionáveis para suas implementações; eles já são construídos em cima de centros de dados distribuídos, replicados e de sincronização flexível. Google e Amazon podem ser considerados como bons exemplos de serviços virtualizados de acesso a dados que foram projetados e implementados com uma previsão de escalabilidade e desempenho.

Integração declarativa

Com aplicativos em escala de Internet, as alianças e colaborações entre organizações são formadas com o uso de contratos digitais. Esses contratos representam o conjunto de acordos no nível de serviço que devem ser estabelecidos para que os trabalhos de computação sejam transmitidos de uma organização para outra, para as origens de dados tornarem-se visíveis, a funcionalidade de equipamentos especiais tornarem-se disponíveis aos parceiros, para o nível de confiança em funções do usuário combinadas, requisitos de segurança e diretivas etc. As organizações virtuais precisam de contratos para controlar suas operações, como é o caso com qualquer colaboração entre empresas no mundo real. Os contratos são lidos, validados e executados por middleware de suporte especializado.

Tabela 1. Serviços/funcionalidades/recursos característicos que os dispositivos podem suportar

Característica	Descrição
Segurança	Todos os aspectos de segurança (p.ex., autenticação, autorização, auditoria, confidencialidade, privacidade, etc.) são abordados usando protocolos interoperativos e de composição como WS-Security, Liberty-Alliance, SAML, XACML, etc. O gerenciamento de identidade e as soluções de federação também precisam estar implementados (p.ex., Active Directory).
Gerenciamento de trabalho	Esses recursos na rede que são capazes de hospedar trabalhos para execução oferecem um serviço de gerenciamento de trabalhos.
Programação	São reunidas informações sobre utilização de recursos e, a seguir, estas são usadas no processo de decisão para a distribuição dos trabalhos para os recursos disponíveis.
Acesso a dados	Esses recursos da rede que fornecem acesso ao armazenamento de dados (p.ex., sistemas de bancos de dados relacionais, sistemas de arquivos, redes de área de armazenamento) precisam apresentar serviços adequados.
Registros de Recursos e monitoramento	Soluções de P2P ou soluções centralizadas precisam ser implantadas para permitir a descoberta e o monitoramento dos dispositivos na rede e seu status.
Gerenciamento de Dispositivo	Os dispositivos na rede podem precisar ser gerenciados remotamente como uma coleção ou individualmente (p.ex., Active Directory, WS-Management).

Tabela 2. Serviços/funcionalidades/recursos característicos que os dispositivos podem suportar

Característica	Descrição
Intermediação	Os serviços que devem agir como intermediários para outros serviços serão implantados para permitir a descoberta dinâmica de recursos ou recursos agregados para permitir o fornecimento de melhor valor.
Pagamento	Será necessária uma infra-estrutura comum para pagamento, semelhante àquela usada atualmente para os pagamentos de cartões de crédito no mundo real.
Serviços lógicos de armazenamento/computacionais	Estarão disponíveis serviços para oferecer acesso aos recursos computacionais e de armazenamento, mesmo se esses recursos não pertencerem a uma única entidade, da mesma forma que existem empresas que oferecem eletricidade no mundo real.
Serviços lógicos de armazenamento/computacionais	Diretórios globais, como o Google, serão necessários para a locação de recursos e outros serviços na Grade. Os domínios de aplicativos podem implantar seus próprios registros especializados como forma de adicionar valor a seus usuários de domínio (p.ex., um registro de serviços relacionados à bioinformática).
Transferência de dados	Quando grandes conjuntos de dados precisam ser transferidos pela Internet, é necessário disponibilizar tecnologias de transferência especializadas e de alto desempenho. A negociação de quais tecnologias de transferência serão usadas ocorrerá com base em protocolos-padrão. Também podem ser empregadas tecnologias P2P.
Contratos e políticas	Vocabulários e softwares de middleware para criar, negociar, executar e monitorar contratos e políticas serão vitais em um ambiente em que são formadas organizações virtuais dinâmicas.
Orquestração	À medida que os serviços são disponibilizados na Internet, serão necessárias tecnologias para orquestrá-las e combiná-las de forma específica aos aplicativos.
Tecnologias relacionadas à semântica	Em um ambiente em que está disponível um amplo número de recursos e serviços, será extremamente importante refletir sobre as informações disponíveis de forma universal.

Além disso, em um mundo digital em que todo tipo de recurso está acessível, os requisitos de aplicativos e as ofertas de serviços são descritos com o uso de linguagens declarativas. O middleware de suporte é responsável por corresponder dinamicamente os requisitos de um aplicativo com uma oferta de serviço. Se necessário, poderá ocorrer negociação dinâmica de acordos de pagamento e nível de serviço.

Por exemplo, um aplicativo pode anunciar que necessita de uma oferta de serviços, recursos de computação com requisitos específicos de hardware e software, um recurso de armazenamento de dados de determinado tamanho, um mecanismo de visualização com um tempo de resposta específico e um serviço de resolução de equações com tempo de operação garantido. Esses requisitos são expressos com o uso de um vocabulário XML. O documento resultante é enviado a um registro (distribuído) e um conjunto de serviços disponíveis são descobertos. O middleware de base negocia o acordo do nível de serviço e os detalhes de pagamento com os serviços resultantes de acordo com os requisitos da solicitação dentro do conjunto de limites que o usuário final estabeleceu. Com o acordo definido, um contrato digital, que pode ser usado em disputas futuras, é assinado por todas as partes envolvidas.

Redes comunitárias

O SETI@home, ClimatePrediction.net e outros projetos semelhantes demonstraram que por meio de redes comunitárias é possível reunir recursos para resolver grandes problemas. Se ignorarmos a controvérsia que cerca o uso de tecnologias P2P para compartilhamento de arquivos, a promessa da computação em grade de colaboração em problemas de negócio e científicos é uma boa correspondência para as capacidades das tecnologias P2P ou redes comunitárias. Futuros aplicativos HPC Internet deverão considerar as tecnologias P2P como tecnologias de ativação de transferências e compartilhamento de arquivos, descoberta de recursos, distribuição de computação etc. Por exemplo, podemos imaginar uma rede P2P que permite que trabalhos sejam enviados e recursos adequados para a execução sejam descobertos automaticamente.

As redes P2P podem ser implementadas com a utilização do mesmo conjunto de tecnologias de Web Services para aproveitar o imenso investimento nos protocolos de qualidade de serviço de base.

O valor comercial

Embora o conceito de computação em grade tenha surgido da comunidade de supercomputadores, as empresas estão percebendo agora o seu valor comercial. Acesso a recursos com pagamento avulso ou com base em assinatura (particularmente recursos de computação de alto desempenho) está começando a surgir como um modelo comercial viável, com grandes empresas já implantando as tecnologias e serviços de ativação. Naturalmente, a integração dessas implantações em aplicativos deve tornar-se onipresente e completamente transparente aos usuários finais para a visão de "computação de utilidade" ou "computação como serviço" tornar-se uma realidade.

Isso produz uma série de oportunidades valiosas. Obviamente, haverá as empresas que serão capazes de colher os benefícios de hospedar recursos de computação econômicos para outras integrarem em seus ambientes de forma ocasional e específica. A recíproca dessa situação é que haverá oportunidades para as empresas planejarem de forma mais eficiente os seus gastos em infra-estrutura de TI e decidir se investimentos antecipados poderão ser substituídos pelo modelo "pague à medida que computar"(ou não), além da agilidade comercial geral que a mudança para uma arquitetura orientada a serviços irá produzir.

Serviços comuns e funcionalidade

As funcionalidades/serviços comuns apresentados antes (Tabela 1) também são necessários em aplicativos HPC em escala de Internet. No entanto, são mais abstratos e devem fazer suposições diferentes sobre o ambiente em que são implantados. Por exemplo, a questão de quem tem permissão de enviar trabalhos para execução no centro de computação de uma organização será definida por meio de contratos digitais, enquanto que a qualidade de serviço que cada interação receberá (alocação de CPU e armazenamento de dados, por exemplo) será controlada pelos acordos de nível de serviço definidos no mesmo contrato. Além da Tabela 1, também observamos o conjunto de serviços/comportamentos comuns da HPC em escala de Internet apresentado na Tabela 2 .

Design e diretrizes de implementação

Tendo discutido em um nível muito alto e abstrato a arquitetura de aplicativos distribuídos de alto desempenho orientados a serviços que podem ser redimensionados através da Internet, iremos fazer agora algumas considerações importantes sobre design e implementação.

Comunicação custa caro

Apesar dos aprimoramentos sempre crescentes na latência e largura de banda das redes, a comunicação através de infra-estruturas de rede comercial é imensamente menos eficiente do que através de interconexões especializadas ou arquiteturas de barramento de memória. Como conseqüência, deve-se tomar cuidado ao desenvolver a arquitetura, fazer o design e construir aplicativos HPC distribuídos de forma a minimizar os custos associados às trocas de mensagens entre os componentes de um aplicativo.

Além dos custos de rede, a comunidade HPC também está preocupada com os custos de computação resultantes do processamento do XML. Porém, esse é um aspecto que está sendo tratado pela comunidade SOAP. De fato, boas implementações do SOAP já abordam o desempenho de mecanismos binários para mensagens curtas [6], o que implica que no final o fator limitante para a transmissão de mensagens em formato binário ou SOAP será a latência e largura de banda da rede. Embora sem pretender denegrir a importância da baixa latência e alta taxa de transferência dos aplicativos HPC, fica claro que o rótulo de lento que o SOAP atraiu é de certo modo merecido.

Princípios e diretivas dos aplicativos orientados a serviços

Acoplamento flexível e escalabilidade são os resultados de design com princípios e arquitetura sensível de software. Adotamos os seguintes princípios ao construir aplicativos orientados a serviços:

- A coleção de protocolos que recebe suporte de um serviço determina a sua semântica comportamental.
- Os serviços ligam-se a mensagens e a informações transportadas através delas e não a estado e pontos de extremidade específicos.

- As mensagens trocadas entre serviços são auto-descritivas (ou seja, como em REST5) na medida em que transportam informações suficientes para permitir que um destinatário estabeleça um contexto de processamento, assim como as informações necessárias para executar a ação desejada.
- Os serviços são implementados e evoluem de forma independente uns dos outros.
- A integração de serviços ocorre por meio de acordos baseados em contratos.

Além dos princípios acima, também incentivamos a seguinte linha de diretrizes ao construir sistemas orientados a serviços:

- **Ausência de estado:** Essa propriedade relaciona-se ao princípio de natureza autodescritiva acima. Os serviços devem ter como alvo trocar mensagens que transportem todas as informações necessárias para receber serviços para restabelecer o contexto de uma interação em uma conversação de várias mensagens. Os serviços sem estado são fáceis de dimensionar e tornam muito simples a tolerância de falha fail-over.
- **Mensagens ricas:** Os custos de comunicação geralmente são altos. Por isso, devemos ter como meta protocolos que envolvam mensagens ricas que resultem em interações grosseiras, minimizando de forma efetiva o número de vezes que um serviço tem que se estender pela rede.
- **Gerenciamento do estado:** De acordo com o design de aplicativos de n camadas tradicional, a implementação de serviço deve delegar todos os aspectos do gerenciamento do estado a armazenamentos de dados dedicados e especializados (como sugerido na Figura 1).
- **Despacho de mensagens:** Não deve haver suposições sobre os mecanismos de despacho empregados pelos serviços. Como resultado, nenhuma informação específica de despacho deve-se vazar das implementações de serviço, atravessar os limites do serviço e ser transportada por meio do conteúdo da mensagem (por exemplo, SOAP-RPC, SOAP estilo RPC , WSDL estilo agrupado por documento ou nomes de métodos transportados como atributos soap:action ou wsa:action).
- **Agrupamento específico de função:** Os arquitetos devem ter em mente que em uma arquitetura orientada a serviços não existem atores como "consumidor", "provedor", "cliente", "servidor" etc. Essas são funções que existem no nível do aplicativo e não nos blocos de construção da arquitetura, os serviços. Em SOAs, existem somente serviços que trocam mensagens. Tratar um par de serviços como cliente/servidor introduz uma forma de agrupamento que pode depois ser difícil de quebrar.

Conclusões

Serviços são uma abstração sensível para encapsular e gerenciar o crescente nível de complexidade dos aplicativos distribuídos. A beleza da orientação a serviços é que os princípios arquitetônicos e as diretrizes são consistentes, de um processo de sistema operacional até um serviço que encapsula o processo comercial inteiro ou até mesmo uma organização inteira. Os requisitos arquitetônicos da computação de alto desempenho em escala de Internet ou " em grade" não são diferentes dos da computação corporativa focada em negócios, por isso princípios e diretrizes idênticos devem ser usados.

Leitura relacionada

1. Open Grid Services Architecture - *http://forge.gridforum.org/projects/ogsa-wg*

2. Globus Toolkit - *http://www.globus.org/toolkit/*

3. OGSA Data Access and Integration - *http://www.ogsadai.org.uk/*

4. Web Services Grid Application Framework - *http://www.neresc.ac.uk/ws-gaf/*

5. Semantic Web - *http://www.w3.org/2001/sw/*

6. W3C Web Services - *http://www.w3.org/2002/ws/*

7. OASIS Web Services - *http://www.oasisopen.org/committees/tc_cat.php?cat=ws*

Agradecimentos

Os autores gostariam de agradecer ao Prof. Paul Watson (School of Computing Science, University of Newcastle upon Tyne) pelos seus valiosos comentários durante a preparação deste artigo.

Notas de rodapé

1. *http://boinc.berkeley.edu*

2. *http://ssdl.org*

3. *http://www.cs.wisc.edu/condor/*

4. *http://www.cs.wisc.edu/condor/birdbath/*

5. *http://mercury.it.swin.edu.au/ctg/AWSA04/Papers/ng.pdf*

6. *http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm*

Sobre os autores

Savas Parastatidis é arquiteto chefe de software da School of Computing Science, da University of Newcastle upon Tyne. *Savas.Parastatidis@newcastle.ac.uk*

Jim Webber é desenvolvedor sênior da ThoughtWorks Australia. *JWebber@thoughtworks.com*

Vinculando Arquitetura e Estratégia de Produto

Charlie Alfred

Introdução

Os sistemas existem para gerar valor para os seus investidores. Infelizmente, às vezes esse ideal é atendido em um grau apenas limitado. Os métodos de desenvolvimento atuais, como cascata, espiral e ágil, freqüentemente fornecem uma orientação incompleta e inadequada para investidores, arquitetos e desenvolvedores.

Este artigo introduz dois conceitos essenciais: modelos de valor e estratégia de arquitetura, que estão faltando em muitos processos de desenvolvimento

A criação de modelos de valor bem definidos fornece uma direção que aprimora a qualidade das decisões de balanceamento (trade-off), particularmente em sistemas que são implantados para muitos usuários em diversos ambientes. A existência de uma estratégia de arquitetura claramente estabelecida fornece uma orientação coerente de alto nível para o sistema, da mesma maneira como a Constituição dos EUA faz para a sua nação. Finalmente, este ensaio irá mostrar como esses dois conceitos podem ser integrados de forma eficaz com os métodos de cascata, espiral ou ágil.

Requisitos são bússolas ineficientes

As nossas maneiras atuais de construir sistemas complexos com uso intensivo de software são ineficientes. Não é a mesma coisa que dizer que são inadequadas. Muitos sistemas construídos com os métodos de cascata e espiral ou ágil são implantados com êxito e são capazes de satisfazer seus investidores. No entanto, muitos não são, e por razões que podem ser corrigidas.

Os processos tradicionais de construção de sistemas com uso intensivo de software, como os métodos de cascata e espiral, dependem de requisitos para fornecer a direção. Uma concepção errada comum é que os requisitos são instruções que descrevem o problema. De acordo com Greenfield e Short [1], não são. Eles definem a solução da perspectiva dos usuários e do patrocinador do sistema.

Os requisitos possuem algumas desvantagens notáveis:

- **Os requisitos normalmente usam uma estrutura binária.** Eles funcionam como passar/ser reprovado de ano em um curso de faculdade e oferecem pouca ajuda ao se tomar decisões de balanceamento. Naturalmente, essas decisões devem ser tomadas em algum ponto do processo. Geralmente elas são tomadas de forma implícita e sem consideração plena das implicações.
- **Os requisitos geralmente são usados como base para especificar critérios testáveis de aceitação de um sistema**. No processo de torná-los específicos, importantes decisões de design são tomadas de forma implícita, sem considerações plenas das implicações. Em algum momento no futuro essas decisões precisarão ser revertidas a um custo significativo ou terminam limitando o potencial do sistema.
- **Os requisitos tendem a tratar da mesma maneira todos os indivíduos de um tipo de usuário determinado**. Por exemplo, os cenários de casos de uso de um sistema médico poderão referir-se a médicos e enfermeiras, enquanto os de um sistema imobiliário poderão referir-se a comprador, vendedor, agente e financiador. O problema é que dois médicos não são iguais e não se satisfazem necessariamente com as mesmas coisas. Existe uma boa razão para os restaurantes populares possuírem muitos itens no cardápio.
- **As informações necessárias para a tomada de decisões efetivas sobre arquitetura de software geralmente não são mencionadas.** Todos os sistemas são implantados em ambientes que colocam obstáculos significativos em seu caminho. Superar esses obstáculos é responsabilidade de todo sistema e alcançar o sucesso apesar deles é a marca de um sistema eficiente. No entanto, a menos que os desenvolvedores possuam um entendimento extremamente profundo do domínio do problema, eles não terão acumulado a perspicácia para fazer bons julgamentos. Ao mesmo tempo, os usuários influentes e os patrocinadores do sistema geralmente possuem essa experiência, mas geralmente não têm o

Tabela 1. **Serviços, funcionalidades e recursos característicos que os dispositivos podem suportar**

Aspecto	Mecanismo	Exemplo
Fornecer recursos úteis	Fornecer uma nova capacidade significativa	Given Imaging, Inc. desenvolveu uma cápsula de 11x26 mm que envolve uma câmara digital que pode passar através do trato gastrintestinal superior do paciente e capturar imagens.
	Melhorar a qualidade das capacidades existentes	A CPU Pentium 4 da Intel usa uma regra de projeto de 90 mm (reduzida de 130mm) e pode realizar 13 bilhões de instruções por segundo.
Superar obstáculos	Tratar os fatores limitantes	Muitos fundos mútuos e portfólios gerenciados de forma privada estão obrigados a enfrentar restrições de investimento. Os sistemas de conformidade pré-transação analisam as transações propostas para verificar que o portfólio continue estando em conformidade.
	Identificar e atenuar riscos	Um sistema de detecção de alterações na direção do vento em um avião comercial detecta a presença de microexplosões que poderiam comprometer um avião no processo de aterrissagem em queda.
Lidar com a mudança	Explorar oportunidades	A eBay reconheceu que a população em rápido crescimento de consumidores com acesso à Internet criou uma oportunidade para fornecer uma capacidade de leilão eletrônico.
	Adaptar-se rapidamente às novas condições	Eastman Kodak reconheceu a virada de tecnologia que habilitou a fotografia digital e conseguiu uma penetração de mercado nesse segmento para compensar as quedas nas vendas de filmes. A Polaroid não teve tanto sucesso com isso.

Métodos ágeis como EXTREME PROGRAMMING e SCRUM fazem uma abordagem ligeiramente diferente. Esses métodos enfatizam algumas alterações úteis, como colaboração íntima entre investidores e desenvolvedores, e iterações de projeto bastante curtas para obter feedback contínuo. A teoria é que a interação entre investidores e desenvolvedores é um mecanismo mais confiável para a navegação do projeto do que um grande investimento inicial em requisitos escritos.

Além disso, os métodos ágeis tendem a favorecer abordagens mais orgânicas e reativas (refactoring) aos que possuem orientação mais determinada (arquitetura). Os proponentes de métodos ágeis falam de deixar a arquitetura de um sistema evoluir. Em algumas situações essa abordagem pode ser eficaz. Um exemplo é quando as necessidades do usuário ou as condições competitivas mudam rapidamente. No entanto, há muitos casos em que essa abordagem pode ser arriscada. Um em particular é quando um produto deve ser desenvolvido para executar em muitos ambientes diferentes e/ou satisfazer investidores com diferentes necessidades e prioridades.

O maior problema com as abordagens cascata, espiral e ágil é que o desenvolvimento do software geralmente se dá sem algumas informações bastante críticas e sem as ferramentas necessárias para reuni-las. Um barco apto para navegar, um rádio que funcione e um conjunto completo de velas são todos necessários, mas não necessariamente o suficiente. Um marinheiro experiente nem pensaria em deixar o porto sem um bom conjunto de mapas náuticos, uma previsão meteorológica de longo alcance e uma maneira confiável de rastrear a localização do barco.

Este ensaio irá discutir dois processos: modelagem de valor e estratégia de arquitetura. Ele mostrará como será o uso eficiente dessas técnicas:

- Capturar informações essenciais sobre o domínio do problema que permita aos usuários e desenvolvedores fazer balanceamentos eficientes.
- Permitir que obstáculos significativos ao êxito sejam identificados e priorizados.
- Permitir que a estratégia de arquitetura seja expressa de uma maneira clara e concisa que possa ser entendida por todos os investidores.

Visão geral do modelo de valor

Sistemas com finalidade determinada são desenvolvidos para criar valor para seus investidores. Na maioria dos casos esse valor é percebido como benéfico porque esses investidores desempenham funções importantes em outros sistemas. Por sua vez, esses outros sistemas existem para criar valor para seus investidores. Essa qualidade recorrente dos sistemas é uma chave na análise e entendimento dos fluxos de valor. A próxima seção (Descobrindo modelos de valor) discute esse ponto com maior profundidade.

A tabela na página anterior apresenta três aspectos vitais de um sistema intencional. São descritos dois mecanismos para alcançar cada aspecto e um exemplo do mundo real é fornecido para cada mecanismo.

Esses três aspectos estão no âmago de um modelo de valor. Para identificá-los e trabalhar com eles de forma mais fácil, precisamos reduzir cada um a uma forma elementar.

A ***Expectativa de valor*** expressa a necessidade de um recurso específico, incluindo o que é fornecido (capacidades), como eles são fornecidos (atributos de qualidade) e como diversos níveis de qualidade são benéficos (função de utilidade). Por exemplo, um motorista pode ter uma expectativa de valor da rapidez e segurança com que o veículo pode parar em uma velocidade de 90 quilômetros por hora.

A ***Força de oposição*** representa alguma força natural ou imposta no ambiente em que um sistema é implantado, que torna satisfatória uma expectativa de valor bem mais difícil. Por exemplo, a eficiência com que um carro pode parar a 90 quilômetros por hora depende do tipo de superfície (asfalto versus cascalho), da inclinação (subida ou descida), das condições (seco, molhado, gelo) e do peso do veículo.

O ***Catalisador de mudança*** representa alguma força ou evento no ambiente que faz com que as expectativas mudem ou que os fatores limitantes tenham um impacto diferente. Por exemplo, a diminuição no custo dos chips de memória e o aumento da densidade de armazenamento tornaram-se um catalisador da fotografia digital.

No restante deste ensaio iremos nos referir às forças de oposição e aos catalisadores de mudança como fatores limitantes e aos três coletivamente como **condutores de valor**.

Para um sistema ser eficiente ao satisfazer os modelos de valor de seus investidores, precisa ser capaz de identificá-los e analisá-los. As abordagens tradicionais, como cenários de casos de uso ou requisitos comerciais/de marketing começam focalizando os tipos de atores com que o sistema interage. Essa abordagem apresenta várias limitações importantes:

Desafios de arquitetura nos sistemas de conformidade pré-transação do portfólio

As regras de conformidade podem especificar os limites superiores e inferiores com o percentual de ativos do portfólio que podem ser investidos em categorias particulares, tais como tipo de segurança, setor da indústria ou região geográfica. Os percentuais de alocação real em um portfólio podem variar com base as alterações de preço, transações e ações corporativas. Se um portfólio não consegue estar em conformidade com suas regras e perde dinheiro, pode-se exigir que indenize seus investidores.

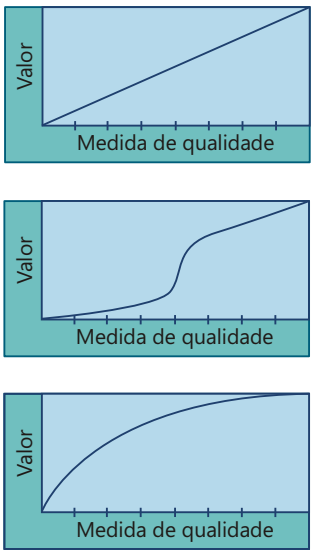
Em situações em que um portfólio está dentro de seus limites de conformidade, há pouco risco de que alguma transação em particular cause uma violação. Entretanto, se o mercado estiver se movendo rapidamente ou se o volume de transações for muito grande, um portfólio que estiver próximo de um de muitos limites tem um risco muito maior de se afastar dos critérios de conformidade.

Um grande desafio é que os mercados de movimento rápido ou os períodos de grande volume de transações constituem precisamente o cenário em que os comerciantes precisam conseguir responder rapidamente para obter os melhores preços. Entretanto, essa pode ser a mesma situação em que há muitas transações pendentes e eventos de alterações de preço para avaliar, tornando o processo de verificação de conformidade mais complexo.

Nesse caso, o tamanho do portfólio, o volume de transações e a volatilidade do mercado, todos se combinam para criar um conflito entre a necessidade de verificação de conformidade (uma técnica de atenuação de risco) e a necessidade de transações eficientes (que impacta o ROI do portfólio). Para serem efetivas, as arquiteturas da organização de gerenciamento do portfólio e seus sistemas de informações devem encontrar uma maneira de equilibrar a compensação entre o risco com a conformidade e transações oportunas.

Figura 1. Curvas de utilidade

Nível do valor	Milhas por galão	Utilidade percebida		
		Linear	Curva S	Parábola
Pior	10 WPG	0	0	0
Adequado	20 WPG	25	10	60
Satisfatório	30 WPG	50	50	80
Preferível	40 WPG	75	85	90
Ótimo	50 WPG	100	100	100



- Concentra-se mais em que coisas os atores fazem e menos em por que elas são feitas.
- Tende a estereotipar os atores em categorias, em que todos os indivíduos de um tipo são essencialmente iguais (comerciantes, gerentes de portfólio ou administradores de sistema, por exemplo).
- Tende a ignorar diferenças nos fatores limitantes (por exemplo: Um investidor em Nova York é o mesmo que um investidor em Londres? Negociar no mercado aberto é o mesmo que negociar durante o dia?).
- Baseia-se em resultados binários: o requisito é atendido ou não. O caso de uso é concluído com êxito ou não.

Existe uma razão bastante lógica e prática de por que essa abordagem é popular. Ela usa um raciocínio seqüencial e baseado em classificação, de modo que é fácil de ensinar e explicar e pode produzir um conjunto de objetivos fáceis de verificar. Naturalmente, se simplicidade fosse o único objetivo que contasse, ainda estaríamos caminhando ou cavalgando para ir de um lugar a outro.

Descobrindo modelos de valor

Em seu livro Competitive Advantage, Michael Porter [7] discute o conceito de cadeias de valores no contexto do planejamento estratégico corporativo:

"Embora as atividades de valor sejam os blocos de construção da vantagem competitiva, a cadeia de valores não é uma coleção de atividades independentes, mas um sistema de atividades interdependentes. Vinculações são os relacionamentos entre a maneira como uma atividade de valor é realizada e o custo ou desempenho de uma outra.

As vinculações existem não apenas dentro da cadeia de valores de uma empresa (vinculações horizontais), mas entre a cadeia de valores da empresa e as cadeias de valores dos fornecedores e canais (vinculações verticais). A maneira como as atividades desse fornecedor ou canal são realizadas afeta o custo ou desempenho das atividades de uma empresa (e vice-versa)."

Se pensarmos em uma empresa (ou uma cadeia de suprimentos) como um sistema e em cada a atividade de valor principal (compra, recebimento, fabricação etc.) como um subsistema, poderemos generalizar a noção de cadeias de valores e vinculações:

- Cada entidade (atividade de valor) possui seu próprio modelo de valor para representar suas expectativas de valor e fatores limitantes.
- Cada vinculação descreve como o modelo de valor de uma entidade se encaixa ao modelo de valor da entidade com a qual está vinculada.
- Cada vinculação entre duas entidades no mesmo sistema é o que Porter denomina de vinculação horizontal. Cada vinculação entre entidades de sistemas diferentes é uma vinculação vertical.

Porter também se refere ao conceito de diferenciação, em que duas entidades que realizam o mesmo conjunto de atividades de valor comportam-se de forma diferente. Um exemplo simples pode ser um táxi versus um ônibus urbano. Embora ambos forneçam transporte terrestre, esses dois contextos possuem recursos diferentes. O ônibus é relativamente barato e segue uma rota e um horário predeterminados. O táxi está disponível sob demanda (exceto quando você realmente precisa dele), opera ponto a ponto, é mais caro e pode conter um número limitado de passageiros. Quando está chovendo, o custo extra de um táxi pode não importar muito.

Questão de Equilíbrio

No restante deste ensaio usaremos o termo **agrupamento de valores** para nos referimos a uma entidade abstrata que realiza um tipo geral de atividade de valor. **Contexto de valor** será usado para nos referimos a uma forma especializada de agrupamento de valores que possua diferenças significativas em expectativas de valores, forças de oposição ou catalisadores de mudança em relação a outros contextos do mesmo agrupamento.

Tanto os agrupamentos de valores quanto os contextos de valor possuem seus próprios modelos de valor. O modelo de valor de um agrupamento representa os aspectos comuns de todos os contextos que especializam esse agrupamento. Cada contexto de valor especializa o modelo de valor do seu agrupamento. O conjunto dos modelos de valor de todos os contextos de um agrupamento fornece uma percepção importante das diferenças entre o que cada um espera e como é afetado pelo seu ambiente.

Por que isso é importante? A arquitetura de um sistema deve realizar um ato de equilíbrio delicado envolvendo seus condutores de valores.

todos os cenários de implantação possuem expectativas de valor e fatores limitantes equivalentes. Provadores de vinho e pilhas AA são bons exemplos de sistemas de contexto único. Outros exemplos são editores de texto simples, analisadores de diferenças de arquivos e muitos outros utilitários de computador. Em um sistema de contexto único ainda é possível haver interdependência e conflitos entre combinações de expectativas de valores e fatores limitantes.

Porém, fica mais desafiador. A maioria dos sistemas complexos possui vários contextos. Em outras palavras, ao se considerar diferentes ambientes de implantação, eles possuem variações significativas nas expectativas de valores, forças de oposição e catalisadores de mudança. À medida que o número de contextos aumenta ou seu grau de compatibilidade diminui, fica muito mais difícil satisfazer todos eles com uma arquitetura única. Embora haja várias técnicas para tratar dessa situação, a primeira etapa é reconhecer quando se está diante dela.

Muitos sistemas possuem apenas alguns contextos. Isso ocorre com maior freqüência com sistemas que são implantados para uso interno em uma organização. Ambientes de implantação diferentes podem possuir fatores limitantes diferentes. Por exemplo, um sistema para despachar manipuladores de bagagem aérea é afetado por extremos climáticos ou um sistema internacional é afetado por regulamentos locais. Outras vezes, os ambientes de implantação possuem expectativas de valores diferentes. Isso é particularmente verdadeiro quando existem diferenças culturais ou internacionais. As enfermeiras que operam máquinas de hemodiálise para pacientes com problemas renais crônicos em um hospital patrocinado pelo governo na Europa irão ter necessidades e prioridades diferentes das enfermeiras que realizam a mesma tarefa em pequenas clínicas particulares nos EUA (onde empresas de seguro privado pagam os tratamentos).

Muitos outros sistemas possuem um grande número de contextos. Eles ocorrem com mais freqüência em produtos centrados em tecnologia, desenvolvidos para venda ou arrendamento a um conjunto amplo de clientes. As mesmas condições que causam variação em sistemas de contexto pequeno, ocorrem em alto grau devido a:

- O número de contextos de implantação pode ser milhares ou milhões de vezes maior,
- As organizações (ou sistemas) dos quais os financiadores participam podem ter conjuntos de expectativas de valores muito diferentes e
- Os catalisadores que acionam mudanças significativas em cada ambiente de implantação provavelmente são muito diferentes.

Em resumo, um modelo de valor captura os condutores que determinam a satisfação de um segmento de mercado específico e como será difícil satisfazê-los.

Curvas de Utilidade

A seção anterior fez referência a um conceito importante chamado curva de utilidade. De modo muito simples, uma curva de utilidade é um mapeamento de uma escala de medição para uma segunda. A primeira escala representa uma variável de resultado que pode ser quantificada. A segunda escala é o nível de valor (satisfação, utilidade) que é gerado. O exemplo mais comum de uma curva de utilidade é a usada para mapear os resultados de provas em notas para exames de colegial ou da faculdade. Como iremos mostrar, um bom entendimento das curvas de utilidade é absolutamente essencial para tomar decisões de balanceamento eficientes.

Figura 2. Formulação da estratégia de arquitetura

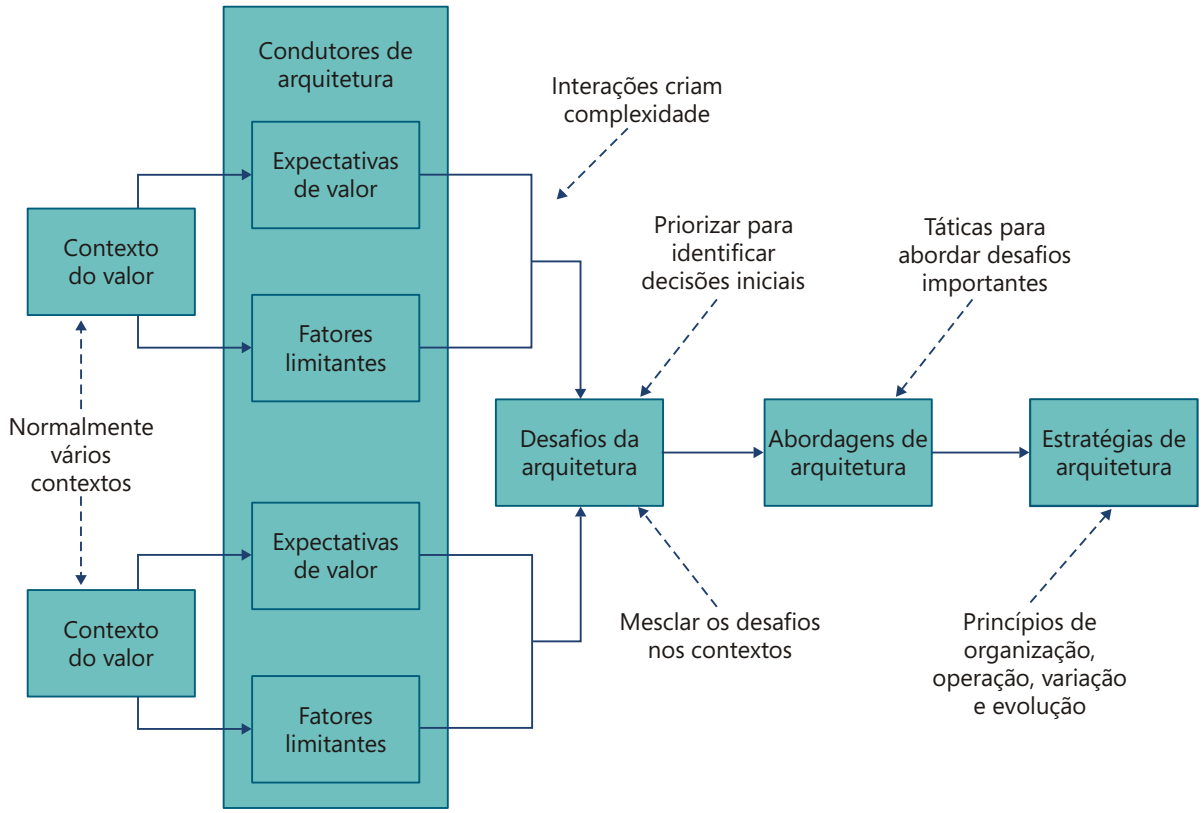
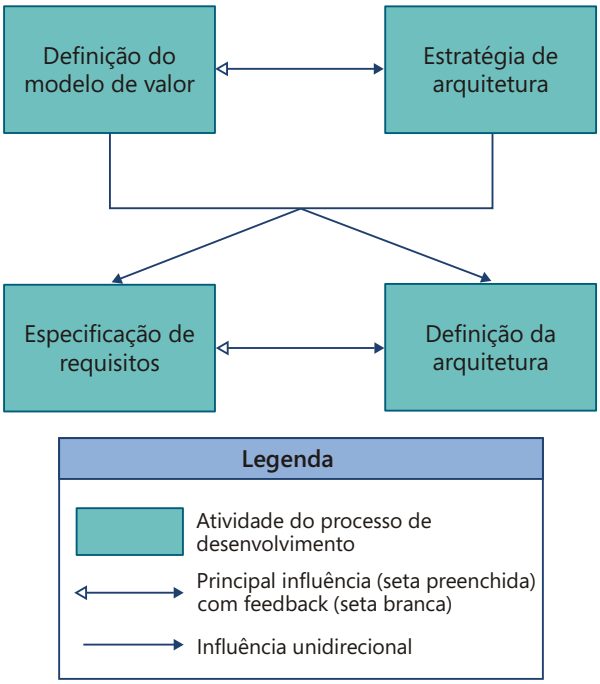


Figura 3. Arquitetura orientada a valor com métodos tradicionais



A Figura 2 ilustra um exemplo simples. A primeira escala representa a economia de combustível combinada na cidade e na estrada para um veículo segundo a Agência de Proteção Ambiental. A segunda escala representa 5 valores qualitativos:

Pior: O requisito mínimo aceitável. Pouco ou nenhum valor é perdido com resultados abaixo desse nível.

Adequado: Esse nível representa um resultado abaixo da média, desapontador mas aceitável.

Satisfatório: Esse é o resultado esperado: nem melhor, nem pior.

Preferível: Esse nível representa um resultado acima da média, satisfatório e compensador, mas não muito acima do comum.

Melhor: O melhor resultado esperado. Pouco ou nenhum valor é ganho com resultados que excedem esse nível.

A figura mostra três curvas de utilidade distintas. Há muitos outros formatos possíveis; esses representam os três mais comuns. A primeira curva é linear, a segunda tem um formato de curva em S e a terceira é uma parábola. Todas as três possuem os mesmos valores exatos pior e melhor. O que é interessante observar são os valores intermediários. Um aumento de 10 a 20 km por litro produz 10% do valor disponível para a curva em S, mas 60% para a parábola.

Em um sistema de contexto único, o uso de curvas de utilidade para analisar estratégias de arquitetura é direto. O método de análise de decisão descrito por Kepner e Tregoe [2] pode ser usado para essa finalidade. Cada alternativa é avaliada em relação a cada expectativa de valor. As curvas de utilidade são usadas para mapear o valor da medida quantitativa obtida por cada alternativa para o seu valor correspondente. Em seguida, os níveis de valor são ponderados pela prioridade da expectativa e totalizados. As alternativas mais preferíveis possuem totais mais altos.

O aspecto mais desafiador desse método é escolher um mecanismo apropriado para avaliar cada alternativa em relação a cada meta desejada. O melhor cenário é quando o mecanismo fornece uma medição objetiva (como medir o consumo ou a potência de um motor de automóvel). Em alguns casos o mecanismo pode ser subjetivo. O custo de produzir uma medição objetiva apropriada deve ser equilibrado pela precisão e objetividade extra obtidas. Em algumas situações, uma avaliação inicial pode ser feita com avaliações subjetivas. Se os resultados forem próximos, medições objetivas podem ser feitas para se escolher entre as melhores alternativas.

Desafios da Arquitetura

Um desafio da arquitetura é uma situação em que um ou mais fatores limitantes tornam mais difícil satisfazer uma ou mais expectativas de valor. Dito de forma simples, um desafio da arquitetura é um obstáculo ou barreira que o sistema deve superar para fornecer valor. Esse é um ponto fundamental. Obstáculos e expectativas de valor são como yin e yang. Se não houver obstáculos presentes o valor cai, porque o resultado é fácil e qualquer um pode fazer. Água engarrafada é uma exceção digna de nota a essa regra.

Dentro de qualquer contexto, a identificação dos desafios da arquitetura envolve avaliar:

- Quais fatores limitantes causam impacto em uma ou mais expectativas de valor?
- Se forem observados impactos, eles tornam a realização das expectativas de valor mais fácil (impacto positivo) ou mais difícil (impacto negativo)?
- Cada impacto torna as coisas mais fáceis ou mais difíceis? Uma escala simples de baixo, médio ou alto geralmente é suficiente?

A Figura 3 descreve alguns desafios da arquitetura que ocorrem em um sistema de conformidade pré-negociação do gerenciamento de portfólio. Uma discussão mais aprofundada dos desafios da arquitetura e um estudo de caso podem ser encontrados em [4].

Os desafios da arquitetura devem ser considerados dentro de seus próprios contextos. Embora possa ser possível tirar uma média das curvas de utilidade através dos contextos, o mesmo não pode ser feito com o impacto de fatores limitantes nas expectativas de valor. Por exemplo, suponha que um servidor da Web forneça páginas a usuários em dois contextos. Um contexto acessa informações estáticas, como documentos de referência. Eles desejam tempos de resposta entre 1 e 3 segundos. O outro contexto acessa informações muito dinâmicas, como resultados de eventos esportivos em andamento. Eles ficam satisfeitos com tempos de resposta na faixa de 3 a 6 segundos.

Os dois contextos estão sujeitos às limitações da CPU, memória, disco e rede . No entanto, à medida que os volumes de solicitação aumentam em um fator de 10 ou 100, esses dois contextos provavelmente irão encontrar obstáculos de redimensionamento bastante diferentes. No caso de conteúdo dinâmico, a sincronização de atualizações e acessos torna-se um fator limitante em situações de carga intensa. Para o conteúdo estático, situações de carga intensa podem ser superadas armazenando em cache as páginas lidas com freqüência.

Existe um ponto final que deve ser mencionado sobre desafios da arquitetura e sistemas com vários contextos. Em muitos casos, pode parecer que um sistema único é capaz de suportar muitos contextos diferentes. No entanto, os contextos de arquitetura que surgem de cada contexto constituem uma boa ferramenta para avaliar a compatibilidade desses contextos um em relação ao outro. Quando contextos incompatíveis são tratados pela mesma arquitetura, o resultado nunca é satisfatório para nenhum deles. Um sofre às custas do outro ou ambos ficam comprometidos. Um exemplo disso é uma ferramenta de semicondutor tentando suportar contextos de produção e de pesquisa com uma única arquitetura. Devido aos conjuntos de expectativas de valor bastante diferentes (confiabilidade versus flexibilidade), forças de oposição (fábrica versus laboratório) e catalisadores de mudança (rodadas de produção versus experimentos), é improvável que esse casamento pudesse ser salvo.

Estratégia de arquitetura

Como as seções anteriores descreveram, a formulação da estratégia de arquitetura de um sistema começa com:

- Reconhecer os contextos de valor apropriados e priorizá-los.
- Definir as curvas de utilidade e priorizar as expectativas de valor em cada contexto.
- Identificar e analisar as forças de oposição e os catalisadores de mudança em cada contexto.
- Detectar onde os fatores limitantes tornam difícil satisfazer as expectativas de valor.

A Figura 2 ilustra esse processo. A lista de atividades anterior nos leva para a caixa Desafios da Arquitetura no meio do diagrama. Neste ponto, estamos trabalhando com uma lista de desafios de arquitetura que foram reunidos de todos os contextos. Cada um desses desafios representa o impacto de um ou mais fatores limitantes em uma ou mais expectativas de valor.

Como o diagrama mostra, antes de começarmos a tratar cada um desses desafios precisamos priorizá-los. As seguintes observações explicam por quê:

- Quanto antes uma decisão for tomada, mais coisas é provável que ela restrinja.
- Quanto mais tarde uma decisão for tomada, menos alternativas estarão disponíveis.

Como resultado, só faz sentido reservar as primeiras decisões de arquitetura para ser as que produzem o maior valor. Existem vários critérios que podem ser usados para priorizar os desafios da arquitetura. Recomendamos um equilíbrio entre o seguinte:

Importância: Qual a altura da prioridade das expectativas de valor que recebem impacto do desafio? Se essas expectativas de valor forem específicas a alguns contextos, qual é prioridade relativa desses contextos?

Magnitude: Qual a intensidade de um impacto nas expectativas de valor que foram causadas pelos fatores limitantes?

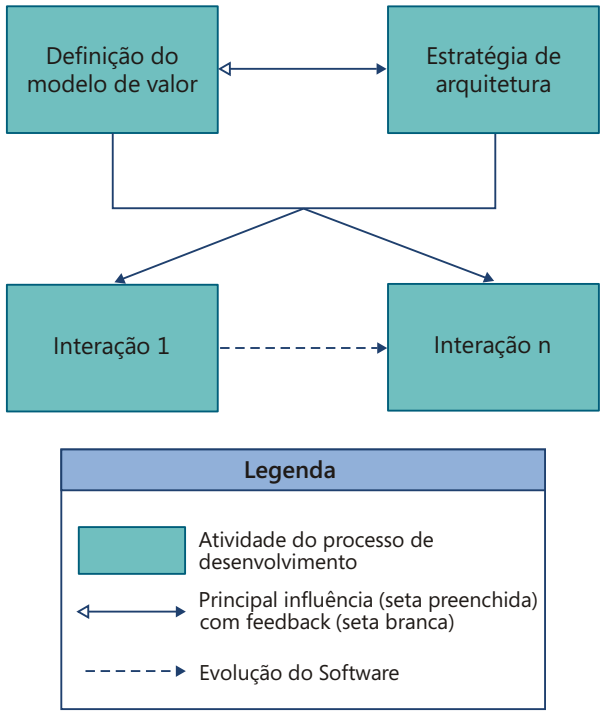
Conseqüência: Quantas opções realistas parece haver? Essas opções apresentam diferenças significativas de dificuldade ou eficiência?

Isolamento: Qual o isolamento do impacto das opções mais realistas? Quanto mais difundido o impacto, mais peso esse fator possui.

Após os desafios da arquitetura estarem priorizados, são formuladas abordagens para os de prioridade mais alta. Embora técnicas como estilos e padrões de arquitetura possam ajudar, essa é uma área em que uma profunda experiência com o problema e os domínios da solução é inestimável. Abordagens eficazes a desafios significativos são resultado de habilidade, percepção, empenho e trabalho compenetrado. Essa afirmação é verdadeira, quer o problema seja cirurgia, gerenciamento executivo ou arquitetura de software.

À medida que cada desafio é tratado, a sua abordagem irá restringir as soluções a outros desafios e, às vezes, criar outros novos. Se as prioridades dos desafios da arquitetura estiverem corretas, a maior parte das restrições em níveis mais específicos também estará apropriada. No entanto, em alguns casos a abordagem de um desafio de alta prioridade poderá ter impacto negativo em vários desafios de prioridade ligeiramente inferior. A prioridade combinada dos desafios que receberam impactos poderá superar o desafio de prioridade superior. Nesse caso, é aconselhável recuar e formular uma abordagem diferente ao desafio original.

Figura 3. Figura 5. Arquitetura orientada a valor com métodos tradicionais



Finalmente, com as abordagens formuladas para o conjunto de desafios de alta prioridade, a estratégia de arquitetura pode ser expressa. O arquiteto analisa o conjunto de abordagens e separa um conjunto de princípios de orientação nas seguintes áreas:

Organização: Como o sistema é organizado em subsistemas e componentes? Qual é a composição e as responsabilidades de cada um? Como o sistema pode ser implantado por uma rede? Que tipos de usuários e sistemas de externos existem? Onde estão localizados e como se conectam?

Operação: Como os componentes interagem? Em que casos a comunicação é síncrona? Em que casos é assíncrona? Como as ações dos componentes são coordenadas? Quando é aceitável configurar um componente ou fazer diagnóstico dele? Como as condições de erro são detectadas, diagnosticadas e corrigidas?

Variabilidade: Que recursos principais do sistema são permitidos variar de um ambiente de implantação para outro ? Que opções são suportadas para cada recurso e quando a opção pode ser feita (por exemplo, em compilação, em vinculação (link), instalação, inicialização ou em tempo de execução)? Que dependências existem entre os pontos de variação?

Evolução: Como o sistema é projetado para dar suporte à mudança enquanto retém a sua estabilidade? Que tipos específicos de mudanças significativas foram antecipados e quais são as maneiras preferidas de abordá-los?

Em resumo, a estratégia de arquitetura é o leme e a quilha de um veleiro, fornecendo direção e estabilidade. Ela deve ser uma breve instrução de direção de alto nível que possa ser entendida por todos os financiadores e deve ser relativamente estável durante o tempo de vida do sistema.

Integração com desenvolvimento existente

Métodos

A Figura 3 mostra como os Modelos de Valor e a Estratégia de arquitetura relacionam-se aos métodos de cascata e espiral. Os Modelos de Valor e a Estratégia de arquitetura operam em um ponto anterior e um nível superior a esses métodos. Quando os modelos de valor são estudados e as estratégias da arquitetura são formuladas, fornecem um alicerces sólido para a especificação de requisitos e a definição de uma arquitetura mais detalhada. O modelo de valor conduz os requisitos e influencia a definição da arquitetura ao fornecer informações para se fazer balanceamentos. A estratégia de arquitetura conduz a definição mais detalhada da arquitetura e fornece um conjunto de requisitos derivados que são necessários para superar obstáculos conhecidos.

Uma analogia apropriada é considerar a estratégia de arquitetura como planejamento estratégico e os modelos de valor como análise de mercado. Sob essa luz, os requisitos tornam-se diretivas e objetivos corporativos. A definição da arquitetura é a organização comercial e o plano operacional, e os casos de uso são o equivalente aos processos comerciais.

Poucas empresas estabelecem objetivos corporativos, estrutura organizacional, planos operacionais e processos comerciais sem primeiro ter uma clara idéia da sua missão, mercados, concorrentes, recursos e estratégia. Até as menos eficientes fazem isso.

A Figura 4 mostra como os Modelos de Valor e a Estratégia de arquitetura relacionam-se aos métodos ágeis. Tanto EXTREME PROGRAMMING quanto SCRUM levam em consideração uma definição da arquitetura. SCRUM faz isso explicitamente, esperando que a arquitetura seja definida nas primeiras 4-5 semanas de iteração.

EXTREME PROGRAMMING faz isso implicitamente. Um dos 12 princípios centrais da EXTREME PROGRAMMING é chamado Metáfora do Sistema.

Esse princípio não é usado com tanta freqüência nem tão bem entendido quanto seus irmãos mais famosos: Pequenos Releases, Programação em Pares, Desenvolvimento Orientado a Teste. Nos primeiros tempos da EXTREME PROGRAMMING, a equipe que trabalhava no grande e complexo Sistema de Folha de Pagamentos da Chrysler precisava de uma boa maneira de descrever o gerenciamento do fluxo de trabalho para os desenvolvedores da Chrysler. Alguém teve a idéia de traçar uma analogia entre o fluxo de trabalho da folha de pagamentos e uma linha de montagem automotiva. A metáfora funcionou e os desenvolvedores da Chrysler conseguiram fazer a imagem.

O site EXTREME PROGRAMMING [6] define a Metáfora do Sistema como o que Extreme Programming (XP) usa em vez de uma arquitetura formal. Uma história simples compartilhada de como o sistema funciona, uma metáfora. Essa história normalmente envolve um punhado de classes e padrões que dão forma ao fluxo central do sistema que está sendo construído.

O que XP chama de "arquitetura formal" é mais próximo daquilo que chamamos acima de definição da arquitetura. Uma estratégia de arquitetura desempenha a mesma função que uma metáfora do sistema, sem ser uma metáfora. Essa é uma vantagem significativa, pois metáforas realmente eficazes (como a usada na Chrysler) podem ser difíceis de conseguir. Em contrapartida, princípios centrais claros e concisos são fáceis de formular e fáceis de entender. Uma pessoa não precisa sair e assistir ao filme Hidalgo para entender o que se quer dizer com "vida, liberdade e a busca da felicidade".

Conclusões

Em resumo, o modelo de valor ajuda-nos a entender e comunicar informações importantes sobre origens de valor. Alguns dos problemas importantes que ele trata são como o valor flui, por que ocorrem similaridades e diferenças nas expectativas de valor e nos fatores externos, e qual subconjunto desse valor o nosso sistema procura satisfazer. É trabalho do arquiteto satisfazer essas expectativas de valor resolvendo forças que influenciam o sistema em geral, forças que são específicas a determinados contextos e forças que se espera que mudem com o tempo. Nesse sentido a arquitetura é semelhante a voar em um avião a jato: o piloto deve transportar os passageiros com segurança a um destino conhecido, enquanto equilibra as leis da aerodinâmica, os recursos do avião e as condições climáticas atuais e futuras. A vinculação entre os modelos de valor e a arquitetura de software é clara e lógica e pode ser expressa pelos nove pontos indicados a seguir:

1. Os produtos e sistemas que fazem uso intenso de software existem para fornecer valor.

2. Valor é uma quantidade escalar que incorpora percepções de utilidade marginal e importância relativa através de muitas metas distintas. Balanceamentos entre metas são uma consideração extremamente importante.

3. O valor existe em vários níveis, alguns dos quais contêm o sistema de destino como um provedor de valor. Os modelos de valor desses escopos contêm os condutores primários da arquitetura de software.

4. Os modelos de valor acima desses na hierarquia podem provocar mudanças nos modelos de valor de seus filhos. Essa é uma entrada importante na formulação dos princípios de evolução do sistema.

5. Para cada agrupamento, os modelos de valor são homogêneos. Os contextos de valor, expostos a diferentes condições ambientais, possuem expectativas de valor diferentes.

6. O patrocinador de desenvolvimento do sistema possui prioridades diferentes para tentar satisfazer diversos contextos de valor.

7. Os desafios da arquitetura resultam do impacto de fatores ambientais nas expectativas de valor em um contexto.

8. As abordagens de arquitetura procuram maximizar o valor tratando primeiro dos desafios de arquitetura de prioridade mais alta.

9. As estratégias de arquitetura são sintetizadas a partir das abordagens de arquitetura de prioridade mais alta decompondo-se as regras, diretivas e princípios comuns de organização, operação, variação e evolução.

As principais contribuições dessa abordagem são:

1. As origens de valor no sistema são modeladas como conceitos de primeira classe. As expectativas de valor associam um pequeno número de recursos a atributos de qualidade, curvas de utilidade e fatores externos. As expectativas de valor são contidas por domínios de valor e contextos; os domínios capturam os aspectos comuns de expectativas de valor, enquanto que os contextos capturam as variabilidades importantes dentro de um domínio.

2. A possibilidade de rastreamento do raciocínio arquitetônico também é uma entidade de primeira classe. As expectativas de valor vinculam-se a desafios da arquitetura, que se vinculam a abordagens de arquitetura, que se vinculam a estratégias de arquitetura. Os investidores podem ver agora o processo de raciocínio que esteve por trás da solução.

3. Um efeito colateral bastante útil dessa capacidade de rastreamento é uma capacidade aumentada de revisar as arquiteturas de software. Como o raciocínio por trás das decisões é tornado explícito, fica mais fácil para outros investidores (patrocinadores do projeto, especialistas em domínio, especialistas em tecnologia, usuários finais) identificar aspectos que possam estar faltando ou que estejam incorretos.

Referências

[1] Greenfield, J. and Short, K., Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools,Wiley, 2004. ISBN: 0-471-2084-3.

[2] Kepner, C., Tregoe, B., The New Rational Manager, Kepner-Tregoe Inc., 1997,

ISBN: 0971562717.

[3] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns: Elements of Reusable Object- Oriented Software, Addison Wesley 1995, ISBN: 0201633612.

[4] Alfred, C., "Using Architecture Challenges to Formulate Software Architecture ", 2002, Foliage Software Systems, Inc. white paper. *http://www.foliage.com/whitepapers/index.shtml*

[5] Kazman, R., Asundi, J., and Klein, M. "Making Architecture Decisions, an Economic Approach", SEI Technical Report: CMU/SEI-2002-TR-35, 2002.

[6] Extreme Programming Core Practices - *http://c2.com/cgi/wiki?ExtremeProgrammingCorePractices*

[7] Porter, M., Competitive Advantage: Creating and Sustaining Superior Performance, MacMillan Inc. 1985, ISBN: 0029250900.

[8] Buschmann, F. et al., Pattern-Oriented Software Architecture, Volume 1: A System of Patterns, 1996, John Wiley and Sons, ISBN: 0471958697.

Sobre o autor

Charles Alfred é diretor técnico da Foliage Software Systems, que alcançou vantagem sobre a concorrência através de estratégia tecnológica, arquitetura de software e desenvolvimento personalizado de softwares. Desde sua fundação em 1991, a Foliage já completou mais de 175 projetos para clientes nas áreas de serviços financeiros, semicondutores, saúde, aviação e e-business.

Notas de rodapé

Uma técnica equivalente para mapear uma métrica quantitativa para uma escala de utilidade está descrita em [5].

Microsoft

THE ARCHITECTURE JOURNAL