Ficha de Caracterização de Trabalho

Título: EAI – Padrões e tecnologias

Resumo: Apresentam-se algumas das tecnologias usadas no complexo mundo da integração de sistemas, procurando sempre que possível recorrer a exemplos para melhor clarificar as vantagens do uso de uma determinada tecnologia em detrimento de outra. É também analisado o middleware que possibilita esta integração de sistemas.

URL: http://student.dei.uc.pt/~tmarto

Data: 30 Outubro 2006

Esforço: 40 Horas

Motivação: Aprofundar os conhecimentos na área da integração de sistemas.

Aprendizagem: Compreensão e análise dos padrões de integração mais utilizados, bem como as problemáticas associadas a cada um deles. Aquisição de conhecimento no modo de comunicação entre sistemas e as camadas de software subadjacentes.

Conteúdos: Integração de sistemas, Information Oriented Application Integration, Service oriented application integration, Portal oriented application integration, Business process integration application integration, Middleware, Message oriented middleware.

Processos: Aprendizagem das melhores técnicas e abordagem de integração de sistemas.

Sequência: Aprofundar o conhecimento nos padrões de integração usando exemplos de casos reais.

EAI – Padrões e Tecnologias

por

Tiago Fonseca Marto

Departamento de Engenharia Informática
Universidade Coimbra
tmarto@student.dei.uc.pt

Resumo: Apresentam-se algumas das tecnologias usadas no complexo mundo da integração de sistemas, procurando sempre que possível recorrer a exemplos para melhor clarificar as vantagens do uso de uma determinada tecnologia em detrimento de outra. É também analisado o middleware que possibilita esta integração de sistemas.

Palavras-chave: Integração de Sistemas, EAI, SOAI, POAI, BPIOAI, Middleware, MOM, mensagens

Introdução

A integração de sistemas de uma ou várias organizações tem sido nos últimos anos o maior objectivo de qualquer organização que pretenda a solidificação num mercado cada vez mais competitivo. Em 2003 um estudo demonstrou que 70% dos projectos de integração falhavam [1] produzindo resultados aquém dos esperados, ultrapassando os orçamentos previstos ou pura e simplesmente não funcionando. Neste artigo são apresentados alguns dos padrões usados actualmente para tentar aumentar a taxa de sucesso dos projectos de integração. Além das diferentes abordagens de integração de sistemas, são também analisadas as vantagens destas e as maiores dificuldades encontradas na sua implementação. No final do artigo será analisada a camada responsável pela comunicação entre aplicações, ou seja, o middleware.

1. Integração de Sistemas

EAI – Enterprise application integration consiste na interligação de vários tipos de software, de vários departamentos de uma organização, ou até mesmo de várias organizações, de modo a partilhar dados e serviços por toda a comunidade. Esta abordagem, apesar de parecer apenas puramente tecnológica, deve ser vista como um valor a acrescentar à organização e deve ser seguida não só pelo departamento de informação, mas também pelo núcleo de decisões de uma empresa, já que esta integração de sistemas só será bem sucedida se forem compreendidos todos os requisitos necessários ao bom funcionamento de uma organização. Além de garantir o bom funcionamento de uma organização, a integração de sistemas pode garantir também a criação de novas cadeias de valor, a redução de custos, a uniformização de processos, etc. [2]

Este conceito está presente sempre que temos dois sistemas de informação e uma rede de comunicação entre ambos, o que é recente é a visão de negócio associada a uma integração de sistemas, por forma a retirar o maior ganho desta partilha de informação e de serviços.

Nos últimos anos a tendência de integração de sistemas tem vindo a passar de uma abordagem orientada pela informação, para uma abordagem orientada por serviços, pois esta permite não só a troca de informação, mas também a partilha de serviços. [3]

Actualmente a integração de sistemas é algo inato a qualquer sistema de desenvolvimento de e-business. As aplicações não podem mais existir sós entre as várias aplicações que compõem o sistema de informação (SI) de uma organização. Pelo contrário, cada aplicação deve partilhar informação para aplicações quer intra, quer extra organizacionais. Esta integração de aplicações baseada apenas na informação é o modo mais popular de conseguir integração de aplicações contudo, outras poderão ser mais vantajosas.

Os dados a integrar são de grande importância. São escassas as empresas que prestam atenção à qualidade dos dados, e aquelas que o fizerem irão certamente beneficiar desta análise, melhorando os processos de negócio. [14]

Ao integrar várias aplicações é também importante ter em atenção qual o grau de integração pretendido, ou seja, se é necessário que as aplicações estejam de tal modo integradas que não seja possível funcionar em separado. Ou por outro lado se se pretende apenas uma simbiose em que as aplicações possam funcionar em separado se indispensável.

Este dilema afecta muitas decisões acerca da escolha da tecnologia certa para uma determinada integração, podendo mesmo ser necessária a escolha de várias tecnologias para obter os melhores resultados.

1.1 Information Oriented Application Integration

1.1.1 Descrição

Esta abordagem defende que a integração de aplicações deve ser feita a nível dos dados, entre as bases de dados das diversas aplicações, isto é, as bases de dados devem ser vistas como o ponto central de toda a integração. Assim existem diversos locais para a captura e actualização destes dados:

- **Base de dados**:
  Neste caso o acesso é feito directamente à base de dados. É o modo mais limpo, simples e rápido de obter dados, obrigando apenas que se tenha acesso à base de dados e que se conheça o seu esquema de dados. Normalmente é feita uma "query" usando SQL (Structured Query Language), que nos devolve um conjunto de resultados. Para fazer actualizações, basta usar o comando SQL de actualizações (update). Apesar de ser o local mais indicado para efectuar a troca de dados, por vezes não temos todo o conhecimento do esquema de base de dados, levando a algumas dúvidas acerca dos resultados.

- **Aplicação**:
  Podendo também ser vista como um serviço, este tipo de integração baseia-se somente nos dados, consistindo numa funcionalidade que o autor da aplicação desenvolveu para fazer consultas de negócio à base de dados. Para clarificar este ponto podemos considerar um sistema de facturação que disponibiliza um serviço de verificação de saldo de um cliente, acessível através de CORBA (Common Object

Request Broker Architecture). Este serviço está encapsulado na aplicação, e não é mais do que uma "query" á base de dados. Se não soubermos como está disposta a informação na base de dados, este serviço torna-se necessário para actualizar a base de dados de destino. Este modo de obtenção de dados, apesar de bastante simples, peca por ser escasso e inflexível, pois na altura de desenvolvimento da aplicação de origem de dados, apenas se cria um API (Application Programming Interface) que responda às perguntas mais genéricas que se possam fazer. Todas as outras terão de ser posteriormente desenvolvidas ou os dados obtidos de outras formas.

- **Interface do utilizador**:

  Esta técnica baseia-se na captação de dados de uma aplicação através da sua interface ao utilizador, isto é, usando uma camada de software que simula o utilizador. Assim é possível extrair dados de uma base de dados fechada ou cujo esquema é desconhecido. Esta técnica é bastante lenta e deve ser apenas usada em ultimo recurso, contudo permite a ligação a software existente acerca do qual nada conhecemos. Em casos de actualizações de software em que a interface seja alterada, toda a integração terá de ser reformulada. Esta técnica é geralmente utilizada na obtenção de dados de sistemas legados, como mainframes. Estes são sistemas fechados em que toda a informação está centrada num ponto e muitas vezes o acesso é feito por terminais comunicando através de uma porta de série. O uso desta técnica deve ser evitado uma vez que o impacto na performance de todo o sistema é bastante grande. Tem a vantagem de permitir comunicar com uma aplicação fechada.

Admitindo que se tem acesso directo à base de dados, a integração pode ser feita de diferentes modos:

- Data replication

  Esta noção de integração consiste na replicação dos dados de uma base de dados para a outra. Para que este cenário possa ocorrer é necessário que ambas as bases de dados tenham mecanismos para receber dados de outra base de dados, e armazenem dados externos como sendo seus, tendo para isso vários esquemas de dados.

  Existem já no mercado camadas de software que são colocadas entre as bases de dados e que permitem a ligação de bases de dados de diferentes vendedores, bem como as transformações de dados de modo a tornar os dados úteis.

- Data Federation

  Este tipo de abordagem obriga a utilização de uma camada de software que faz a ligação da aplicação com as várias bases de dados existentes. Por exemplo se tivermos uma base de dados em cada departamento, este middleware irá esconder as várias bases de dados, de modo a que aplicação apenas veja uma base de dados virtual. Este middleware é responsável por conhecer exactamente o esquema de cada uma das bases de dados.

1.1.2 Vantagens

Uma vez que estamos a lidar apenas com dados, não é necessária a alteração de origem e de destino nas aplicações já existentes, uma vez que todos os motores de bases de dados já têm incorporados procedimentos para o tratamento de dados.

O facto de lidarmos apenas com dados, permite-nos esquecer o funcionamento da aplicação, a lógica do negócio e outras questões mais complexas.

Visto que as alterações são apenas ao nível da camada de dados a aplicação necessita de poucas ou nenhumas alterações.

Esta técnica é acima de tudo mais fácil e barata de utilizar quando comparada com outras tecnologias, havendo bastantes ferramentas já disponíveis no mercado, caso sejam necessárias. [3]

Uma vez que as aplicações se mantêm inalteradas não é necessário dar uma nova formação aos empregados, que continuam a trabalhar do mesmo modo.

1.1.3 Desvantagens

Uma das desvantagens desta abordagem está relacionada com as actualizações do software. Por exemplo, caso o esquema de dados do CRM (Customer Relationship Management) seja alterado, então toda a integração deixa de funcionar.

Esta abordagem será considerada inútil se for necessário adicionar algum tipo de acção à informação. Um exemplo que permite ver esta limitação é a gestão de stocks. Sempre que o número de existências de uma determinada peça está abaixo de uma quota é necessário efectuar uma nova encomenda. Com esta abordagem, a base de dados sabe que o número de existências está abaixo da quota predefinida, contudo não pode fazer nada para contrariar este facto.

1.1.4 Implementação

Apesar de parecer simples, a aplicação desta técnica está longe de o ser. Antes de mais é necessária a análise de todos os esquemas das bases de dados, o que implica saber a que dizem respeito todas as colunas de todas as tabelas. Caso esta análise seja mal feita, então corremos o sério risco de fornecer informações erradas ao utilizador, bem como destabilizar todo o sistema.

De seguida é necessário que a equipa de desenvolvimento determine quais as necessidades a nível de frequência das actualizações. Se for uma base de dados relativamente estática, a actualização pode ser feita apenas diariamente. Contudo, se as actualizações forem bastante frequentes e a necessidade de manter as bases de dados actualizadas for de crítica importância, então terá de ser implementado um sistema de actualizações em tempo real, o que aumenta a dificuldade e os custos de integração.

Depois de determinados os dados a copiar e a frequência de actualizações, são escolhidos os modos de transferência de dados. Estes podem ser apenas updates em tabelas remotas ou então query's efectuadas por uma camada de software abstracta, com vista a obter as alterações desde a última actualização. Esta camada abstracta seria depois responsável também por actualizar a base de dados de destino, com os dados já transformados.

Caso se opte pela criação de uma base de dados virtual (data federation) será necessário proceder à actualização da origem de dados nas aplicações.

1.2 Service oriented application integration

1.2.1 Descrição

Esta é a abordagem mais cara, e mais invasiva de todas as abordagens de integração de aplicações revistas aqui [3], contudo os resultados podem não ser os desejados. Para evitar um mau investimento é necessário ter em conta vários factores, tais como a reutilização de serviços, a real necessidade desta abordagem, os custos, e a

capacidade da empresa aguentar todas as dificuldades até que se comecem a obter resultados positivos.

Service oriented application integration (SOAI) permite às empresas partilharem não só informação como também serviços. Esta partilha de serviços pode ser feita através da disponibilização dos mesmos num servidor central, ou através do uso de aplicações que invocam serviços remotos.

No âmbito da partilha de serviços, em 1991 foi lançado o CORBA, e desde então este conceito tem vindo a evoluir [4], levando ao aparecimento de enumeras tecnologias e standards, culminado neste momento no advento dos web-services de segunda geração [3].

Durante vários anos, estes servidores que forneciam o acesso a serviços tinham apenas em vista uma das grandes vantagens de uma abordagem orientada a serviços: a reutilização de código. Agora estes serviços são vistos como excelentes pontos de integração de sistemas, reduzindo assim o tempo de integração devido à replicação de serviços redundantes.

O principal objectivo de SOAI é conseguir nivelar todos os serviços remotos usando algum tipo de camada aplicacional responsável pela invocação destes serviços na máquina remota, como se se tratassem de serviços locais. O resultado final é uma aplicação composta por vários serviços remotos, podendo estes estar em diferentes locais. Para exemplificar, podemos considerar a emissão de facturas que faz uso de serviços do departamento de contabilidade, do departamento de vendas, e do armazém.

1.2.2 Vantagens

Uma abordagem orientada a serviços permite que a empresa possa mais facilmente mudar alguns dos seus modelos de negócios, respondendo assim mais rapidamente à mudança de área de negócio, ou à integração de uma nova área de negócio numa empresa já existente.

Apesar de a maior parte dos serviços terem sido desenvolvidos a pensar numa utilização intra-organizacional, é possível disponibilizar estes serviços para fora da organização, aumentando assim o seu valor no mercado, tanto para o cliente final (serviço de calculo de custos de envio), como para outras empresas (B2B).

Apesar da reutilização de serviços se apresentar como uma grande vantagem, segundo a consultora Gartner, apenas 10 a 40 % dos serviços são reutilizados [5]. Este facto prende-se não só com a diferença de processos entre as várias organizações, mas também por políticas internas que tentam esconder o seu modo de funcionamento para evitar cópias e assim poder ganhar maior quota de mercado.

Com o nivelamento dos serviços através de um middleware é possível criar um service bus, composto por serviços provenientes de várias empresas associadas.

1.2.3 Desvantagens

Ao contrário de outras abordagens, esta implica a alteração de algumas, senão mesmo todas as aplicações existentes no SI, de forma a acomodar os novos serviços. O uso de web-services é uma das vertentes que as organizações que basearam o seu SI nesta abordagem, pretendem tirar proveito num futuro próximo. Todavia alguns destes SI's, apesar de orientados a serviços, terão de sofrer grandes alterações, ou em alguns casos serem reconstruídos de raiz. [3] Este facto faz com que esta abordagem seja a mais cara e intrusiva.

1.2.4 Implementação

Ao integrar diferentes aplicações de acordo com esta abordagem, é frequente o uso de middleware para esconder as diferenças tecnológicas dos vários sistemas que compõem o SI. Por exemplo, perante um departamento com serviços em c++ a correr em Windows NT e outro com serviços em Java sobre uma maquina Linux, podem ser seguidos 2 caminhos. Ou é colocada toda a lógica aplicacional num servidor partilhado entre as aplicações, ou é necessário alterar toda a aplicação usando um mecanismo de partilha de serviços, como Distributed Object Technology (Corba, COM+, etc), para a criação de aplicações "tightly coupled" que permitem o acesso a todos os serviços. Qualquer uma destas soluções permite a ligação dos diversos serviços entre si, e posteriormente a colocação dos mesmos à disposição não só da empresa como também para fora desta (web-services).

1.3 Portal oriented application integration

1.3.1 Descrição

Os portais não são propriamente uma tecnologia nova. Eles existem desde o nascimento da Internet e compreende-se o valor da divulgação de dados empresariais através de interfaces web. Este foi o nascimento de B2C e-business, que levou á criação dos B2B e-business, que é o cerne da questão da abordagem orientada a portais. Actualmente cada vez mais empresas deixam de usar o fax e o telefone para fazer encomendas ou solicitar serviços, fazendo isto apenas através da interacção com um web-browser. Infelizmente, apesar de serem uma necessidade, os portais não suportam um dos objectivos da integração de aplicações: o tempo real. Esta necessidade não esta presente em algumas actividades de negócio, contudo em negócios fortemente voláteis, como o mercado de acções, o tempo real é uma necessidade.

Esta técnica de integração de sistemas B2B necessita de um utilizador que use o seu web-browser, ou então um sistema de integração que simule um utilizador, para receber e enviar informação. Esta técnica foi anteriormente abordada em IOAI.

Imaginemos então o seguinte caso. Um cliente pretende ver informação relativa às suas facturas. O cliente acede pelo browser ao servidor web, que terá de ser capaz de juntar a informação disponível numa base de dados SAP do departamento de contabilidade, e também de uma base de dados ORACLE que contém os dados relativos ao cliente como nome, morada etc. Para conseguir isto, o application server que está a servir o web-server está configurado para estabelecer uma ligação às bases de dados das aplicações departamentais.

1.3.2 Vantagens

A principal vantagem desta abordagem prende-se com o facto de não ser necessário efectuar alterações no back-office para garantir que este consegue comunicar com outro sistema de uma outra empresa. Esta comunicação é feita somente através de uma camada interna, web-server+application server+middleware, diminuindo assim os riscos inerentes à alteração da aplicação, bem como os custo de desenvolvimento de testes dessas alterações.

Outra vantagem é a segurança. Com esta abordagem não é necessária a configuração de firewall para permitir o uso por parte de entidades forasteiras. Todo o

tráfego de chegada é tratado pelo web-server, ferramenta esta que devido à sua maturidade é já bastante segura.

A versatilidade apresentada pelos sistemas desenvolvidos de acordo com esta abordagem é também notória, pois transforma um SI fechado sobre si mesmo, num SI com uma grande capacidade de virar para o exterior e interagir com outros agentes.

Os custos do desenvolvimento desta integração são bastante mais baixos e a sua implementação mais rápida.

1.3.3 Desvantagens

A informação não corre livremente, é necessária a intervenção do factor humano. Como resultado disto a informação não é automaticamente actualizada, há sempre uma latência resultante da interacção com o utilizador.

Abstracção da informação. Como a informação tem de passar por canais que são independentes da lógica da organização, é necessário transformar estes dados, adicionando assim complexidade á solução.

Privacidade. Uma vez que os dados estão a ser publicados para toda a web não existe garantia de protecção contra um vasto leque de ataques.

1.3.4 Implementação

Toda a integração segundo esta abordagem é efectuada antes de chegar ao servidor web.

Para que a informação possa ser devidamente acedida, a aplicação (interface do utilizador e comportamento da aplicação) tem de ser programada de acordo com os dados existentes nas diversas aplicações do SI interno. Apesar de existirem enumeras tecnologias para o desenvolvimento de portais, a mais usada nesta abordagem é a "application servers". A escolha desta tecnologia prende-se com o facto de existirem já no mercado várias ferramentas que possibilitam a interacção das aplicações mais comuns (SAP, ORACLE, etc.). Apesar de não integrar a aplicação directamente, esta abordagem fornece ao utilizador dados de diversas aplicações, bem como a possibilidade de fazer actualizações nas mesmas. O padrão de integração que possibilita esta interacção com outras aplicações é normalmente o IOAI.

1.4 Business process integration application integration

1.4.1 Descrição

BPIOAI é a ciência e o mecanismo que gere o movimento de dados e a invocação de serviços de aplicação de um modo correcto, de modo a suportar a gestão e organização de processos comuns que existem dentro e entre organizações, bem como dentro da própria aplicação. Esta tecnologia fornece assim uma nova camada de criação de processos que pode abranger diferentes aplicações em diferentes organizações.

Esta abordagem vê o middleware como a canalização, isto é, os diversos canais de comunicação com as várias aplicações, e considera-as como dados adquiridos, prontos a funcionar.

Apesar da imaturidade das ferramentas existentes, esta abordagem deve ser vista como o passo seguinte numa lógica de integração de sistemas. Isto é de tal modo verdade que vários vendedores de software promovem o seu sistema de BPIOAI como sendo a parte vital de qualquer sistema de integração de sistemas.

Algumas destas ferramentas apresentam já alguns processos de negócio comuns a muitas actividades (por ex. uma venda ou efectuar uma encomenda), contudo esta abordagem não se preocupa em optimizar a velocidade de acesso, ou até mesmo a escalabilidade da solução, caso se tratem de sistemas separados. Este deverá ser o objectivo das camadas inferiores de comunicação, pois o verdadeiro objectivo de BPIOAI é automatizar o movimento de dados e processos de negócio [6], para que outra camada de BPIOAI possa ser desenvolvida reutilizando os processos de negócio já definidos. Por outras palavras, BPIOAI completa a integração de aplicações permitindo não só a partilha de informação e serviços, como também gerir a partilha destes.

Assim, esta abordagem pode ser vista como o futuro da integração de sistemas, pois é basicamente uma técnica que usa outros padrões de integração como camadas inferiores (SOAI, IOAI, e ate mesmo POAI), conseguindo mesmo uma total abstracção das tecnologias responsáveis pelo seu funcionamento. [3]

1.4.2 Vantagens

Com este modelo é possível uma melhor adaptação do SI às necessidades inerentes ao modelo de negócio. Permite também efectuar reajustes caso se verifiquem alterações no modo de funcionamento desejado.

1.4.3 Desvantagens

Ao adicionar uma camada extra ao nosso processo de integração é de esperar um aumento da complexidade na execução do mesmo. Assim, esta abordagem está dependente da aplicação de outras técnicas de integração.

1.4.4 Implementação

Ao integrar sistemas de acordo com esta abordagem, é vantajoso o isolamento de cada acção na aplicação, isto é, remove-la de toda e qualquer sequência da aplicação. O uso do sistema de routing em softwares de BPIOAI permite que qualquer informação necessária seja extraída de qualquer fonte, seja ela uma base de dados, aplicação ou até mesmo um serviço. Deste modo, quando procuramos alterar o processo de negócio apenas precisamos de alterar o modelo desenvolvido e não todas as formas de interacção com as diversas fontes de informação. Esta decisão de implementação permite ainda a reutilização de fontes ou destinos entre os vários processos de negócio.

Apesar de parecer simples, a integração de aplicações usando esta abordagem apresenta algumas dificuldades na configuração do middleware de comunicação com as fontes/destinos de informação.

Todos os processos de negócio existentes nas empresas a integrar deverão ser cuidadosamente documentados, de modo a permitir uma maior compreensão dos processos e dados envolvidos, e os que realmente são necessários. Este é um tipo de acção que deve ser tomada em qualquer tipo de abordagem de integração de sistemas, mas nesta abordagem esta documentação tem um peso ainda maior.

Os processos em falta que são gerados devido à integração das aplicações devem ser bem analisados, de modo a distinguir quais os dados e processos realmente necessários, para que se possa definir um processo de negócio eficiente, evitando ao mínimo posteriores alterações. [6]

2. Middleware

2.1 Descrição

Apesar de não ser um padrão de integração de sistemas, o middleware é responsável a um nível inferior por garantir a ligação de vários sistemas de informação diferentes. Esta camada depende não só da abordagem de integração escolhida, como também das tecnologias existentes nos meios a integrar.

De uma forma simples, middleware é qualquer tipo de software que permite a comunicação entre dois ou mais sistemas.

De facto o middleware pode ser visto apenas como um "pipe" no sistema operativo que permite a comunicação entre dois processos, mas também pode ser um completo sistema de troca de mensagens com vários mecanismos integrados para nos darem garantias e fiabilidade de entrega de mensagens.

Há algum tempo atrás o middleware era negligenciado, sendo visto apenas como uma simples ferramenta que servia de transporte de dados; contudo cada vez mais esta ferramenta tem vindo a ser desenvolvida, resolvendo uma boa parte dos problemas associados à integração de sistemas entre várias empresas. Ao longo dos tempos, esta evolução fez com que o middleware deixasse de ser uma ferramenta de uso interno, passando assim a ser também responsável pela comunicação externa da organização [3].

2.1.1 Middleware ponto-a-ponto

Este tipo de comunicação usa apenas um canal que estabelece a ligação do ponto A para o ponto B. Quando a aplicação A pretende enviar informação, basta coloca-la no canal, e a aplicação B recebe-a. Esta operação é bastante simples, não sendo necessário encapsular a mensagem num formato especial para que esta chegue ao seu destino. Quando comparado com outros tipos de middleware, a sua maior desvantagem prende-se com a impossibilidade de ligar com vários sistemas ao mesmo tempo, além de não ter ferramentas que possibilitem a alteração das mensagens em trânsito, de modo a facilitar uma integração de dois sistemas diferentes.

A maior dificuldade em integrar sistemas com ligações ponto-a-ponto está na complexidade em manter todos os sistemas em constante comunicação, especialmente quando tempos um número razoável de sistemas a integrar. Com a chegada de um novo sistema para integrar é necessário alterar todos os outros, de modo a criar um novo canal de comunicação entre todas as aplicações já existentes e a nova. Se o número de sistemas a ligar for baixo, a simplicidade deste tipo de abordagem pode ser o factor dominante na escolha de um middleware.

Como exemplos deste tipo de comunicação temos tecnologias baseadas em RPC's e alguns MOM's.

2.1.2 Middleware multi-ponto

Como o nome indica, podemos ter várias aplicações a comunicar com várias outras aplicações ao mesmo tempo. Esta possibilidade faz com que seja a melhor opção para a maior parte dos sistemas a integrar e o seu uso tem vindo a aumentar nos últimos anos. Devido à sua maior capacidade, este tipo de middleware é capaz de suportar um maior número de funcionalidades que facilitam também a integração de vários sistemas.

Apesar de facilitar bastante o trabalho de integração depois de implementado, esta implementação do middleware é bastante mais complexa do que uma abordagem ponto-a-ponto, devendo por isso ser vista como um investimento que só mais tarde irá compensar.

Tecnologias que usem este tipo de abordagem são alguns MOM's (mais complexos), application servers, etc.

2.1.3 Síncrono / assíncrono

A comunicação feita entre o middleware e as aplicações a integrar pode ser feita de acordo com dois modelos:

- modelo síncrono - tem de estar fortemente integrado (tightly coupled) para que possa funcionar, tem de assumir que ambas as aplicações a comunicar estão a funcionar e disponíveis, e que a ligação entre ambos está estabelecida. Uma vez que a comunicação é síncrona, para o middleware entregar uma mensagem tem de bloquear e esperar pela resposta a devolver ao sistema que iniciou a troca de mensagens.
- modelo assíncrono - apresenta enormes vantagens no caso especifico da integração de sistemas. Este modelo não faz quaisquer assumpções acerca dos sistemas que estão a comunicar, limitando-se a receber uma mensagem do emissor e tentar entrega-la ao receptor assim que este esteja disponível.

Com o advento da integração de sistemas, cada vez mais empresas de software procuram a criação de ferramentas que sirvam melhor muitos dos problemas de integração dos diversos sistemas.

Assim, quando se está num meio heterogéneo e em que nada pode ser assumido acerca de sistemas remotos, o melhor modo de conseguir ter sucesso é seguir uma abordagem "loosely coupled" e usar um sistema baseado em mensagens. Deste modo o programador não tem a preocupação de verificar se a mensagem foi entregue, qual a resposta, etc. Ao usar comunicação assíncrona, o sistema não fica vulnerável ao lag introduzido pela rede e pode continuar o seu processamento, sendo contudo necessário programar um modulo responsável pela recepção de mensagens de resposta.

2.2 Message oriented Middleware

2.2.1 Descrição

Neste tipo de middleware toda a comunicação é baseada em mensagens. A gestão de mensagens é normalmente efectuada por um software designado por MOM – Message Oriented Middleware. Um sistema de mensagens gere todo o tráfego de mensagens de acordo com as configurações pré-definidas para cada aplicação. Para cada uma deve ser definido um canal, os tradutores de mensagens, os destinos possíveis e os tipos de mensagens. O administrador tem ainda ao seu cargo a criação de uma metodologia para tratar mensagens que não foram entregues ou que foram rejeitadas. Após toda esta configuração, o sistema deve ser capaz de lidar com todo o envio de mensagens, entre vários sistemas heterogéneos, com segurança e fiabilidade. [8]

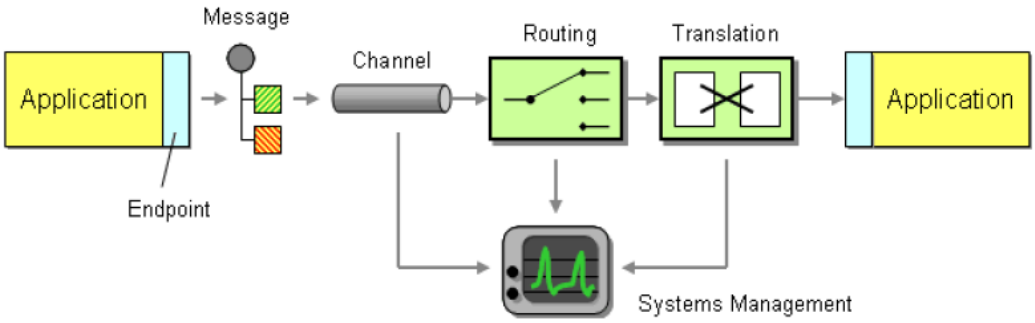

Figura 1 – Modo de funcionamento básico de um sistema de mensagens

A razão pela qual um sistema de mensagens é necessário prende-se com o facto das redes que ligam os diversos sistemas serem invariavelmente pouco fiáveis. Porque uma aplicação está pronta para enviar dados, isso não significa que o receptor esteja pronto a receber essa mensagem, e mesmo que ambas as aplicações estejam preparadas a rede pode estar "em baixo". Para vencer esta dificuldade surgiu o MOM que fica responsável pela entrega da mensagem de um modo assíncrono.

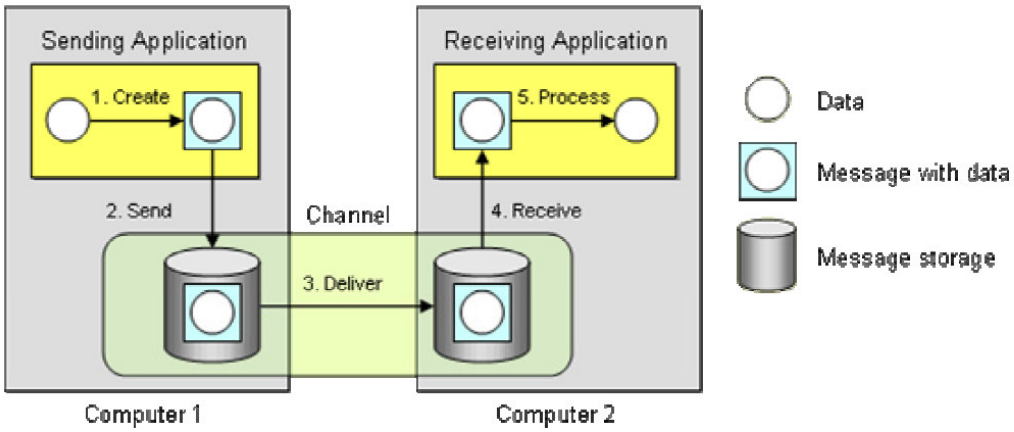

Figura 2 – Processo de envio de uma mensagem

A figura 2 mostra dois importantes conceitos:

1. "Send and forget" – Como podemos constatar no passo 2 a aplicação emissora coloca a mensagem no canal. Assim que o envio terminou o emissor pode continuar a trabalhar, enquanto que o MOM se encarrega de entregar a mensagem ao destinatário. O emissor pode então assumir que o receptor irá receber a mensagem eventualmente, e não tem de ficar à espera que isto suceda. [8]

2. "Store-and-forward" – No passo 2, quando a aplicação coloca a mensagem no canal, o MOM guarda esta mensagem no disco do emissor. No passo 3 o MOM entrega a mensagem ao computador receptor, escrevendo-a em primeiro lugar no disco e posteriormente entregando-a á aplicação de destino. Este processo, apesar de parecer penoso, garante que a mensagem é enviada em casos de falha de rede do lado do emissor e em casos de erro por parte do programa de destino no receptor. [8]

2.2.2 Vantagens

Comunicação remota:

O uso de mensagens permite que aplicações separadas comuniquem entre si e transfiram dados. Se tivermos 2 objectos que residam no mesmo processo, estes podem estar simplesmente em memoria, mas se estes estiverem em máquinas diferentes, então é muito mais complexo efectuar esta comunicação. Este processo implica que os objectos possam ser "serializable", isto é, que possam ser convertidos num simples "byte stream", e que possam ser enviados através de uma rede. Caso a comunicação entre maquinas diferentes não seja necessária, então podemos apenas usar memória partilhada entre aplicações.

Integração de plataformas:

Quando estamos a interligar vários sistemas de informação é frequente depararmo-nos com diferentes linguagens, tecnologias e plataformas, que muito provavelmente foram desenvolvidos por empresas diferentes ao longo dos tempos. Para interligar todo este novelo de software é necessária a criação de uma camada responsável pela comunicação, e que seja independente da linguagem ou plataforma de destino. Assim o MOM pode servir como tradutor universal entre as aplicações, convertendo a informação para um formato neutro (p.ex.XML) e no fim traduzindo para a linguagem do destinatário. Esta conectividade universal é o cerne de um "message bus".

Comunicação assíncrona:

As mensagens permitem o uso de uma abordagem "send and forget". O emissor não tem de esperar por uma resposta do receptor, e assim pode mais rapidamente continuar com a sua execução. Devido a este modo de funcionamento, é necessário que o programa emissor esteja preparado para receber mensagens de callback para eventuais respostas ou "aknowledgements".

Escalonamento:

Com o uso de comunicação síncrona, o processo invocador tem de esperar que o processo invocado termine a computação de resultados e a resposta deste. Mesmo admitindo que tudo corra bem, o tempo necessário à computação de uma resposta pode demorar bastante tempo, ficando assim o sistema bloqueado à espera de uma resposta. Considerando um caso de actualização de salários de uma empresa com uns milhares de empregados, esta operação de alterar cada salário 1 a 1, iria demorar bastante tempo. Com o uso de comunicação assíncrona é possível o processo emissor enviar todos os pedidos seguidos, não tendo de estar à espera de nenhuma resposta, efectuando estes pedidos o mais rápido que consegue. A velocidade a que estes pedidos seriam tratados seria também muito maior, pois trataria todos estes pedidos sem interrupções.

Sobrecarga:

Com o uso de RPC's (Remote Procedure Calls), se muitas chamadas forem feitas ao mesmo receptor, este corre o risco de ficar sobrecarregado e deixar de ter capacidade para responder a novos pedidos, comprometendo assim o funcionamento de

uma organização. Com o uso de mensagens é o processo receptor que controla a velocidade de tratamento das mensagens recebidas, evitando assim uma sobrecarga.

Fiabilidade:

Devido ao "Store and forward", a comunicação torna-se extremamente fiável capaz de tolerar não só as falhas de comunicação, como também algumas falhas de software, em caso de "crash" do programa. Por exemplo, no caso de o emissor não conseguir estabelecer comunicação com o destinatário, o MOM guarda a mensagem em disco e tenta enviar a mensagem até conseguir. O receptor, assim que obtém ligação, recebe as mensagens, guarda-as em disco, e assim que a aplicação receptor é ligada, entrega-as. Caso esta entrega não seja possível a mensagem é guardada em disco, para posterior entrega.

Funcionamento offline:

Algumas aplicações, devido à sua natureza foram concebidas para trabalhar offline, tais como PDA ou portáteis. Neste casos em que é necessária mobilidade e a conectividade é bastante limitada, o uso de mensagens é ideal para garantir o bom funcionamento das aplicações, pois o processo de sincronização é feito apenas com a recepção e o envio das mensagens criadas ao longo do dia.

Mediação:

Com o uso de MOM não é necessário estabelecer ligações com os vários programas com que se pretende comunicar, o que seria muito mais falível. O MOM assume o papel de mediador de comunicação: um ponto centralizado por onde toda a comunicação passa, permitindo analisar os dados enviados para que possam ser tomadas medidas para aumentar a qualidade de serviço, tais como o uso de recursos redundantes e o balanceamento de carga. Exemplo: o MOM pode ser usado para melhorar o acesso a uma base de dados com um elevado número de ligações.

Gestão de Threads:

A comunicação assíncrona evita que a thread invocadora fique bloqueada à espera de resposta. No caso de um servidor que faça muitos pedidos, isso iria significar um numero exageradamente grande de threads bloqueadas a concorrerem pelo mesmo recurso. Usando o MOM, basta que apenas algumas threads estejam à escuta de chamadas "callback" para garantir o bom funcionamento do SI. Estas threads seriam então responsáveis pelo tratamento dessa mensagem de resposta. Um menor número de threads bloqueadas significa também que a recuperação em caso de crash é mais fácil.

Maturidade:

Devido ao seu extenso uso e relativa idade no mercado, há já bastantes ferramentas e de grande qualidade no mercado. Este facto possibilita a sua utilização em grandes organizações, pois são já ferramentas maduras e estáveis, cuja qualidade tem vindo a ser comprovada ao longo dos anos.

Transacções:

No caso de uma operação ser formada por várias mensagens, seria possível que ocorressem erros na execução da mesma caso uma das mensagens chegasse fora de ordem. Para evitar este tipo de situação, o MOM permite o uso de transacções. Sempre que o emissor desejar estabelecer uma comunicação deste tipo basta enviar uma

mensagem primeiro, indicando que irá iniciar uma transacção, envia as mensagens necessárias e no fim efectua um "commit" para sinalizar o fim da transacção. Este modo de operação permite que uma operação só seja executada se estiverem presentes todas as mensagens necessárias e de acordo com a ordem enviada.

Simplicidade:

Ao usar esta abordagem, o programador deixa de ser responsável pela gestão de possíveis quebras de rede, ficando estas a cargo do "midleware" de entrega de mensagens. Em casos de quebra de conexão o sistema de entrega de mensagens continua a tentar enviar a mensagem até que tenha sucesso.

Directórios

Este tipo de abordagem permite que a mesma mensagem seja replicada para vários receptores, apenas emitindo uma mensagem. Este caso é especialmente útil quando temos uma organização composta por vários departamentos e cada um deles tem o seu software de CRM específico. Ao usar esta funcionalidade, é possível apenas com uma mensagem alterar, por exemplo, a morada de um cliente para que não haja problemas a nível do negócio, como enviar a mercadoria para uma morada (departamento de shipping) e enviar a conta para outra morada (departamento de billing).

2.2.3 Desvantagens

A comunicação assíncrona não é a panaceia da integração de sistemas. Ela resolve muitos dos desafios inerentes à interligação de um sistema heterogéneo de uma forma elegante, mas como todas as soluções, cria novos problemas. Alguns destes problemas são inerentes ao modelo de comunicação, enquanto que outros advêm de cada caso específico.

Complexidade:

Uma vez que o sistema não é baseado em chamada de métodos, é necessário que as mensagens recebidas pelo callback sejam tratadas. Este tipo de abordagem torna-se bastante complexo se tivermos muitas mensagens diferentes, pois a thread responsável por isto tem de saber exactamente o que fazer com cada mensagem. Este tipo de abordagem torna a mitigação de erros mais difícil, pois a chamada de uma execução não tem resposta imediata, esta será apenas entregue posteriormente, mas sem previsão de tempo de chegada.

Cenários síncronos:

Nem todas as aplicações podem funcionar de acordo com esta abordagem. Se por exemplo um utilizador estiver a consultar uma base de dados, ele deseja ver os dados rapidamente, e não eventualmente no futuro. O nosso SI deverá então estar preparado para esta eventualidade.

Performance:

O sistema de gestão de mensagens pode afectar o desempenho da aplicação, uma vez que este, para efectuar as comunicações, adiciona algum overhead às mensagens, e precisa de algum tempo de processador para fazer a conversão de e para XML (caso seja esta a linguagem adoptada para o envio de mensagens).

No caso de replicação de uma base de dados, o sistema de mensagens pode ser usado, mas não é uma boa solução, pois irá criar um enorme número de mensagens que irão precisar de um grande tempo de conversão de e para XML. Para este tipo de situações é preferível o uso de ferramentas ETL especializadas. O serviço de mensagens é ideal para manter depois as bases de dados sincronizadas.

Integração

Alguns sistemas de mensagens são fechados, dificultando a integração com outros sistemas de mensagens.

2.2.4 Soluções comerciais

Uma vez que a integração de sistemas é uma excelente oportunidade de negócio, há um grande numero de produtores que vende este midleware ou que o incorpora como parte da sua solução quer seja um sistema operativo, uma base de dados ou outro pacote de software de integração de sistemas. A própria Microsoft incluiu no Windows XP/2000 o seu Microsoft Message Queuing. Este software é acessível através de API's e componentes com. A Oracle inclui também o Oracle AQ juntamente com o seu motor de base de dados. A SUN incluiu o JMS no seu "application server" e desde então todos os "application servers" baseados em J2EE incluem o JMS. [3]

Há cada vez mais empresas que se especializaram em fazer a ligação entre web services e um MOM, permitindo assim uma melhor integração dos diversos sistemas de informação. [7]

Conclusões

"…, it's not about the products, it's not about the technology. A lot of it is about how you go about putting together the structure to make these things all work." [1]

Neste artigo foram analisados diversos tipos de abordagens para integrar vários sistemas de informação, contudo não há um sistema que seja capaz de resolver todas as questões, sendo por vezes necessário usar várias abordagens ao mesmo tempo para chegar a resultados positivos.

A integração de sistemas é um tema bastante actual que acompanhará sempre cada implementação de software e cada SI. [9]

Referencias

[1] Gian Trotta, "Dancing Around EAI 'Bear Traps'", 2003 from http://www.ebizq.net/topics/int_sbp/features/3463.html

[2] Ankur Laroia, "How EAI Can Improve Profitability in the Energy Industry", 2003 from http://www.ebizq.net/topics/eai/features/1634.html?&pp=1

[3] Linthicum, David S., "Next Generation Application Integration", Addison-Wesley, 2004

[4] "History of CORBA", from http://www.omg.org/gettingstarted/history_of_corba.htm

[5] Christopher Koch, "The ABC's of SOA", from http://www.cio.com/abcs/soa_abc.html Gartnet ...

[6] Trividh Patel, "Using Process Centric Approach to achieve Organizational Agility" http://eai.ittoolbox.com/pub/TP060603.pdf

[7] Cummins, Fred A., "*Enterprise Integration"*, John Wiley& Sons, 2002

[8] Gregor Hohpe e Bobby Woolf, "*Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*", Addison-Wesley Professional , 2003

[9] Glen Kunene, "Enterprise Application Integration: The Problem that Won't Go Away" from http://www.devx.com/enterprise/Article/10667

[10] Chris Britton, Peter Bye, "*IT Architectures and Middleware: Strategies for Building Large, Integrated Systems*", Addison-Wesley, 2004

[11] Enterprise Application Integration from http://en.wikipedia.org/wiki/Enterprise_application_integration

[12] Allan E. Alter, "Are Enterprise Apps On the Way Out?" from http://www.cioinsight.com/article2/0,1540,2033113,00.asp

[13] Microsoft, "Guidelines for Application Integration – patterns & practices", Microsoft, 2003

[14] T. Friedman, "A Strategic Approach to Improving Data Quality", Gartner research, 2002