CAPÍTULO 21

Middleware para sistemas colaborativos

José Maria Nazar David
Rita Suzana Pitangueira Maciel

META

Apresentar os conceitos de middleware como uma infraestrutura de apoio à construção e execução de sistemas colaborativos.

OBJETIVOS EDUCACIONAIS

Após o estudo desse capítulo, você deverá ser capaz de:

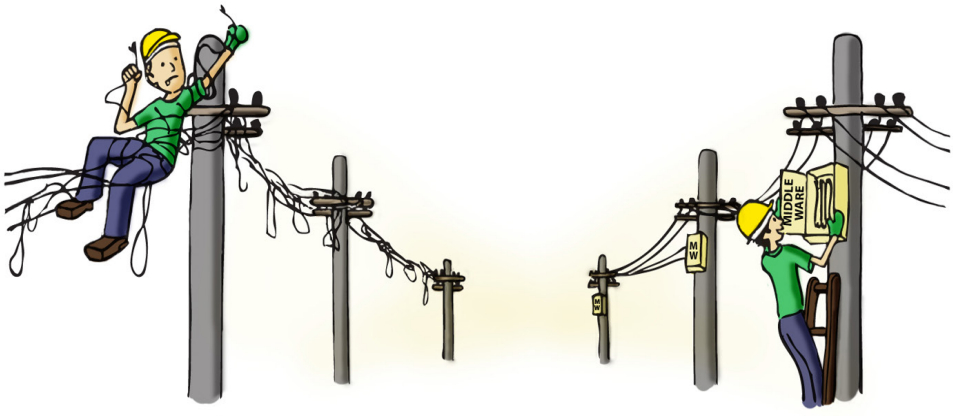
- Definir as características de um sistema colaborativo que levam à adoção de uma infraestrutura de middleware.
- Selecionar uma plataforma de middleware de acordo com as necessidades do desenvolvedor.
- Identificar os principais elementos de um middleware para apoiar o desenvolvimento de sistemas colaborativos.

RESUMO

As plataformas de middleware auxiliam no desenvolvimento de sistemas colaborativos distribuídos. Nesse capítulo são apresentados os motivos para a adoção de plataformas de middleware para a construção de sistemas distribuídos, em especial sistemas colaborativos. São discutidos os conceitos e os requisitos para a definição de uma infraestrutura. Dentre os requisitos apresentados, dois foram detalhados: a capacidade de interoperação dos sistemas distribuídos e dos serviços; e o suporte oferecido pela infraestrutura para a integração de serviços. Foram apresentados os motivos para a adoção de plataformas de middleware para apoiar a integração de sistemas colaborativos, mantendo o foco nas suas principais funcionalidades.

21.1 Middleware

Desde o advento das redes de computadores locais e com a popularização e uso massivo da internet nos mais diversos setores da sociedade moderna, sistemas distribuídos têm sido implementados, comumente, com o uso de middleware. O termo middleware tem sido utilizado em diversos contextos da ciência da computação, e neste capítulo será visto no contexto do desenvolvimento e execução dos sistemas colaborativos. Sistemas colaborativos geralmente são sistemas distribuídos, com membros do grupo dispersos no espaço e no tempo, o que implica em funcionalidades mais complexas que exigem mais esforços dos desenvolvedores. Middleware é um software para facilitar o desenvolvimento e execução de sistemas distribuídos. Consiste numa infraestrutura para dar suporte a diversas características desejáveis para a implementação dos sistemas colaborativos: interoperabilidade, integração, portabilidade, escalabilidade e suporte a diferentes modos de colaboração.



Quando um sistema colaborativo é construído sem o suporte de uma infraestrutura adequada, esforços de implementações de mais baixo nível são necessários para a implementação de requisitos funcionais e não funcionais, como segurança, nomeação e localização. Apesar de a diversidade de arquiteturas e de componentes para o desenvolvimento e execução de sistemas colaborativos, muito ainda precisa ser feito para reduzir o esforço do desenvolvimento e, consequentemente, também diminuir o custo e o tempo de desenvolvimento.

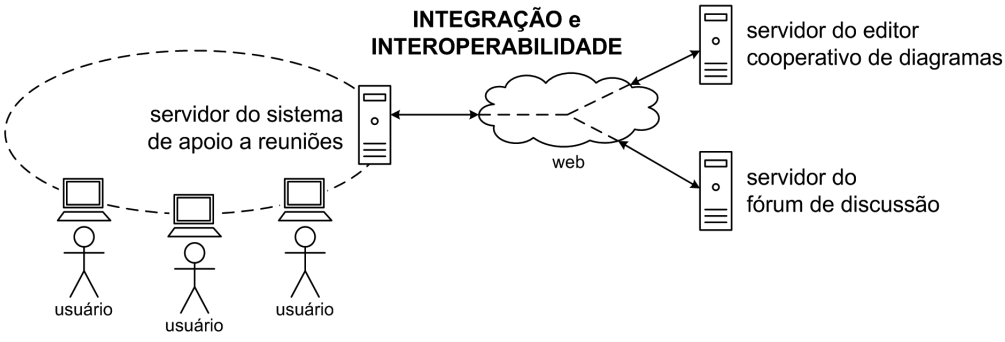


Figura 21.1 Middleware para integração e interoperabilidade de sistemas colaborativos

Para ilustrar o desenvolvimento de sistemas colaborativos, consideremos o cenário ilustrado pela Figura 21.1. Podemos considerar duas situações: o desenvolvimento de um sistema para apoiar a interação dos participantes de um grupo; e a construção de serviços desenvolvidos para serem integrados a diferentes sistemas. Suponha que o grupo de usuários necessite interagir por meio de um sistema de suporte a reuniões. Um sistema colaborativo é inicialmente projetado de acordo com os requisitos dos usuários desse grupo. Entretanto, este sistema pode evoluir para integrar alguns serviços existentes que ampliam o suporte às atividades do grupo. Com a integração de serviços ao sistema de reuniões, os usuários também poderão, por exemplo, discutir e editar diagramas. Entretanto, a integração de serviços é um processo que adiciona complexidade e esforço adicional ao desenvolvimento do sistema.

O suporte para o desenvolvimento de sistemas colaborativos vai além do suporte para a comunicação de sistemas distribuídos. O suporte se estende à integração de sistemas em uma mesma máquina, bem como à interoperabilidade entre sistemas em máquinas distintas. No contexto deste capítulo, interoperabilidade se refere à capacidade dos sistemas trocarem dados e realizarem procedimentos com independência em relação à manutenção; e integração se refere à capacidade dos sistemas trabalharem conjuntamente.

Para apoiar as atividades de construção de sistemas colaborativos, algumas soluções fornecem serviços isolados que implementam funcionalidades para tratar as complexidades desses sistemas. Estes serviços isolados não "conversam" uns com os outros, e por isso nem sempre oferecerem a flexibilidade necessária para apoiar adequadamente a construção de sistemas colaborativos. Quando os sistemas colaborativos não estão integrados e não possuem a capacidade de interoperar, os usuários utilizam sistemas isolados. Além do esforço necessário para a colaboração, os usuários precisarão gerenciar as tarefas e os dados entre os sistemas. Consideremos, por exemplo, a atividade de edição cooperativa de um texto. Mesmo que os participantes utilizem o mesmo editor, deverão estabelecer formas para controlar as versões produzidas, caso contrário algumas contribuições serão perdidas ocasionando retrabalho. As mensagens trocadas durante o processo de edição deverão ser armazenadas em um local comum para a posterior recuperação das decisões tomadas. Mecanismos de coordenação também deverão ser previamente acordados entre os autores, relacionados, por exemplo, ao momento em que cada um pode adicionar contribuições ao texto compartilhado.

Middleware apoia a integração e interoperabilidade dos sistemas com os serviços. Provê aos desenvolvedores de sistemas a facilidade de utilização e localização dos serviços, e apoia o desenvolvimento de serviços com funções que possibilitem a comunicação com os sistemas, mesmo que implementados em diferentes linguagens de programação, de tal forma que a integridade dos sistemas seja mantida. Um middleware também controla processos executados em máquinas diferentes, oferece suporte para a localização e nomeação dos recursos distribuídos, e controla a consistência dos dados distribuídos.

O middleware apoia o desenvolvimento de sistemas colaborativos com tecnologias que promovem a escalabilidade, a robustez e a disponibilidade. Muitos sistemas são construídos para serem executados em um único servidor. Esta arquitetura coloca em risco a disponibilidade dos serviços, pois quando ocorre uma falha do servidor único, todos os serviços ficam indisponíveis. Para sistemas críticos, que exigem continuidade na execução dos serviços, é necessário utilizar algum mecanismo de tolerância a falhas, como por exemplo, um servidor secundário para substituição do servidor primário em caso de falha.

Serviços projetados para o reuso são construídos sem que os requisitos do sistema tenham sido previamente definidos, e devem evoluir para atender às especificidades de cada sistema, por exemplo, com relação à cultura e às políticas de trabalho de um grupo. É fundamental oferecer uma infraestrutura que apoie a flexibilidade e a escalabilidade dos serviços para o suporte ao desenvolvimento de sistemas colaborativos. O middleware oferece facilidades para apoiar diferentes modalidades de colaboração (síncrona, assíncrona e quase síncrona), de interfaces, de linguagens de programação e de banco de dados.

Middleware também oferece flexibilidade para apoiar a execução de sistemas colaborativos entre diferentes plataformas de hardware e software. O ambiente de execução dos sistemas distribuídos apresenta grande heterogeneidade decorrente dos seguintes elementos:

- Plataformas de hardware: desktop, servidores, celulares.
- Tecnologia de rede: local, longa distância, wireless.
- Sistemas operacionais: Windows, MAC Os, Unix, Androide.
- Linguagens de programação: Java, C, PHP.

A heterogeneidade torna o desenvolvimento dos sistemas distribuídos uma tarefa não trivial. Posicionadas entre o sistema operacional de um computador e os sistemas que rodam sob este sistema operacional, plataformas de middleware tratam a heterogeneidade, o que facilita o trabalho do desenvolvedor de sistemas. Facilidades incluem, por exemplo, o uso de operações de objetos distribuídos em máquinas distintas como se estivessem em uma só máquina. Passa a ser tarefa do middleware, e não do programador, localizar o endereço do objeto e executar a operação, responsabilizando-se pela entrega dos valores de entrada e de saída. Quando uma plataforma de middleware não é adotada, os programadores de sistemas distribuídos têm que usar primitivas de baixo nível dos sistemas operacionais e da rede (ex: sockets, pipes, fila de mensagens, memória compartilhada, entre outras). Quando um middleware está intermediando a comunicação entre os sistemas e o sistema operacional e rede, primitivas de mais alto nível podem ser usadas, tal como uma chamada de um método em um programa escrito na linguagem Java. Estas primitivas são chamadas primitivas de interação do middleware e definem o modelo de interação.

Ao ocultar detalhes de implementação relacionados à programação de mais baixo nível, a utilização de um middleware agiliza o desenvolvimento de sistemas colaborativos, reduz a complexidade e, consequentemente, diminui o custo e o tempo de desenvolvimento. A atenção e o esforço de desenvolvimento passam a estar adequadamente direcionados para o tratamento dos requisitos funcionais do sistema colaborativo.

Para favorecer o desenvolvimento de sistemas colaborativos, um middleware deve prover:

- Mecanismos para suporte à interação remota. Diversos elementos do sistema devem interagir mesmo que estejam localizados em uma mesma máquina ou em máquinas distintas.
- Transparência de distribuição. A forma como o sistema requisita funcionalidades deve ser a mesma para elementos locais ou distribuídos. Uma infraestrutura de middleware pode oferecer transparência de acesso, localização, migração, replicação,

falhas, concorrência, entre outros aspectos que devem ser tratados na distribuição dos elementos.

- Independência de tecnologia. Os elementos de um sistema devem interagir mesmo que implementados em tecnologias distintas.

Plataformas de middleware disponibilizam uma API (Application Program Interface) para os sistemas usarem os serviços do middleware. Diferentes sistemas que usam um middleware comum tornam-se interoperáveis, pois o middleware é usado como um elemento que realiza a intermediação da troca de dados e procedimentos.

Middleware provê mecanismos para que o programador perceba o ambiente como um todo e não como um conjunto independente de recursos. Ao utilizar middleware, programadores de sistemas colaborativos são beneficiados com o requisito não funcional da transparência, que está relacionada aos seguintes aspectos:

- Localização: recursos devem ser acessados sem a necessidade do conhecimento da localização física.
- Acesso: se refere à unicidade das operações de acesso a recursos locais e remotos.
- Concorrência: diversos processos acessam recursos compartilhados de forma concorrente sem interferências entre eles.

DESVANTAGENS DE MIDDLEWARE

Middleware distintos não são prontamente interoperáveis devido à ausência de padronização dos serviços oferecidos. Mesmo as plataformas de middleware que interoperam em redes locais não são capazes de interoperar através da internet em função de mecanismos de segurança como firewalls. Neste caso é necessário o uso de procedimentos adicionais a esta infraestrutura para que os sistemas que utilizam esses middleware se tornem interoperáveis. Como resultado, o custo para o desenvolvimento aumenta.

Para sistemas cujo desempenho é um fator crítico, o uso de middleware deve ser avaliado cuidadosamente. Por ser uma camada intermediária entre os sistemas, o sistema operacional e a rede, ocorre um aumento no tempo de resposta dos diferentes serviços oferecidos pelo middleware.

21.2 Categorias de middleware

Um middleware é categorizado em função de diferentes características: tipo de comunicação, linguagens para construção dos sistemas, ambientes de execução, entre outras. Embora não exista uma categorização padrão, uma bastante usada é baseada no mecanismo que o middleware disponibiliza para que seus serviços se comuniquem, denominado primitiva de interação. Os componentes de um sistema, que podem estar dispersos na rede e implementados em tecnologias distintas e heterogêneas, precisam se comunicar para a realização das tarefas, trocam mensagens entre si, disponibilizam e requisitam funcionalidades. Por exemplo, um editor cooperativo de texto necessita verificar se determinado autor possui permissão para

alterar um documento. Neste momento, o editor se torna cliente do sistema servidor do banco de dados que armazena as informações sobre permissões de edição. Diversas mensagens podem ser trocadas entre o editor e o banco de dados para realizar a verificação. Em função da primitiva de interação para estabelecer a comunicação, o middleware é classificado em: procedural, orientado a objeto, orientado a mensagem, ou transacional.

Middleware procedural

Um middleware procedural implementa o modelo cliente/servidor e usa como primitiva de interação RPC (Remote Procedure Call – Chamada Remota de Procedimento). RPC é baseada nas chamadas a procedimento presentes nas linguagens procedurais, como C e Pascal, porém uma RPC possibilita a chamada de procedimentos remotamente, ou seja, localizados em outras máquinas na rede. Uma RPC possui uma interface que define um conjunto de procedimentos e parâmetros de entrada e saída. A interface está escrita em uma linguagem específica de definição de interfaces (IDL – Interface Definition Language, Linguagem para Definição de Interface), que é independente da linguagem utilizada nos programas que implementam o procedimento. Um processo cliente chama procedimentos definidos nas interfaces e implementados por um processo servidor. A comunicação é geralmente implementada de forma síncrona, de modo que o cliente permanece bloqueado à espera da resposta do servidor, comumente chamado de protocolo requisição/resposta (request/reply ou request/wait for replay). Para resolver a heterogeneidade na representação de dados entre as plataformas do cliente, além de realizar a chamada aos procedimentos remotos, um middleware procedural é responsável também por garantir que os parâmetros dos procedimentos sejam passados do cliente para o servidor em um formato de dados homogêneo. Quando um procedimento é chamado, é realizada uma conversão do formato de dados do processo cliente para o formato comum que é conhecido pelos dois processos envolvidos. Esta conversão é chamada de serialização (marshalling). Ao chegar ao processo servidor, os dados devem ser convertidos para o formato específico do servidor. Este processo é chamado de desserialização (unmarshalling). Middleware procedurais são muito usados para implementar sistemas cliente/servidor.

Middleware orientado a objeto

Middleware orientado a objeto é uma evolução do middleware procedural. Objetos disponibilizam métodos, descritos por uma Linguagem para Definição de Interface, que podem ser chamados por outros objetos. Para solicitar a execução de um método de objeto é necessária uma referência que forneça a localização exata deste objeto na rede e no computador em que está implementado. Um middleware orientado a objeto viabiliza a comunicação entre objetos distribuídos e heterogêneos por meio de um serviço para obter as referências das operações, bem como funcionalidades de serialização e desserialização. Além do suporte à comunicação síncrona, esse tipo de middleware também possibilita a comunicação assíncrona entre os objetos.

Middleware orientado a mensagem

Middleware orientado a mensagem dá suporte à comunicação entre os componentes de um sistema distribuído por meio da passagem de mensagem. Uma mensagem pode ser a notificação da ocorrência de um evento ou uma solicitação de execução de uma operação.

Enquanto nos middleware procedural e orientado a objetos o foco é a comunicação síncrona entre dois elementos (um para um), o middleware orientado a mensagem torna usual a comunicação assíncrona e em grupo (um para muitos). Sistemas colaborativos distintos podem trocar mensagens por meio de um fila de mensagens ou pela notificação de eventos. No modelo de interação por meio de uma fila de mensagens, um sistema A coloca a mensagem em uma fila disponibilizada pelo sistema B, e a mensagem aguarda até chegar a sua vez de ser processada por B. Neste modelo não há bloqueio de sistemas enquanto as mensagens são trocadas. As primitivas são send/receive (envio/recepção). Os sistemas também podem registrar interesse em receber mensagens originadas por outros sistemas. Ao gerar um evento, um sistema notifica para o middleware que se encarrega de distribuir para os sistemas que se inscreveram como interessados. As primitivas são: publish/subscribe (publicar/inscrever).

Middleware transacional (MT)

Middleware transacional, também conhecido como Monitor de Processamento de Transações, apoia a execução de transações distribuídas. Fornece suporte à coordenação e sincronização para a execução de transações que necessitam acessar bases de dados diversas. A primitiva de interação é uma combinação de RPC associada a um controle de transação que implementa o protocolo "two phase commit". Neste protocolo, uma transação é realizada em duas fases. Na primeira assegura-se a disponibilidade dos recursos necessários para a realização da transação. Na segunda fase, são realizados os comandos necessários para efetivação da transação. Caso os recursos não estejam disponíveis, ou alguma das bases distribuídas não consiga realizar uma das fases, a transação não é efetivada. Este tipo de middleware é usado por sistemas que demandam rapidez na execução de transações remotas em bancos de dados distribuídos.

PLATAFORMAS DE MIDDLEWARE

Uma plataforma de middleware pode implementar um ou mais tipos de middleware. Plataformas proveem um ambiente de programação onde os sistemas são desenvolvidos, e um ambiente de execução para que os sistemas desempenhem as funcionalidades de forma distribuída.

Diversas plataformas de middleware foram implementadas por diferentes fabricantes. Common Object Request Broker (CORBA) da OMG (Object Management Group), Java Remote Method Invocation (RMI) e Enterprise Java Beans (EJB) da Sun são exemplos de middleware orientado a objeto. Java Message Service (JMS) da Sun, MQSeries da IBM e MSQM da Microsoft são exemplos de middleware orientado a mensagem. São exemplos de middleware transacional: Java Transaction Service, JOTM (Java Open Transaction Manager) e o BEA da Tuxedo. Alguns implementam mais de uma primitiva de interação, como exemplificam Distributed Computing Environment (DCE) da Open Software Foundation (OSF), e Open Network Computing (ONC) da Sun.

Um middleware tem diferentes unidades de implementação que se comunicam utilizando as primitivas de interação da plataforma. Em um middleware procedural, as unidades são os elementos que disponibilizam ou requisitam funcionalidades por meio do modelo cliente/

servidor. Já em middleware orientados a objetos, as unidades são os objetos distribuídos. Na plataforma CORBA CCM as unidades são componentes.

Além da taxonomia relacionada às primitivas de interação, middleware podem ser classificados em relação à dependência de linguagem, padronização, tipo de comunicação, entre outras. A classificação em relação à dependência de linguagem diz respeito à diversidade de linguagens de programação usadas no desenvolvimento dos serviços do middleware. Sob este ponto de vista, middleware podem ser classificados como dependente ou independente de linguagem. Middleware dependentes de linguagem assumem explicitamente a adoção de apenas uma linguagem de programação para o desenvolvimento dos serviços, por exemplo, EJB, Java RMI, Jini. Middleware independentes de linguagem possibilitam que os serviços sejam desenvolvidos em linguagens de programação distintas. Comumente adotam a estratégia de escrita da interface dos serviços em uma IDL. Desta forma, os serviços realizam mapeamentos dos comandos especificados nesta IDL para o formato de cada linguagem de programação.

Em relação à padronização, middleware podem ser baseados em padrões abertos ou podem ser proprietários. Os middleware baseados em padrões abertos seguem normas estabelecidas por um determinado padrão especificado por uma determinada organização. Por exemplo, o CORBA segue o padrão OMA (Object Management Architecture) da OMG. A adesão a padrões tem como objetivo alcançar a interoperabilidade entre plataformas distintas de middleware que implementam o mesmo padrão. Um middleware proprietário segue uma solução específica de um fabricante, protegendo assim a propriedade intelectual ou comercial dos fabricantes.

CORBA É PADRÃO DE IMPLEMENTAÇÃO DE MIDDLEWARE?

Uma tentativa para padronizar a implementação de middleware foi proposta pela OMG na especificação do CORBA. Apesar de ser uma tentativa para se estabelecer uma referência em relação à definição de vários elementos de um middleware, a especificação ainda não se tornou um padrão de fato.

Considerando o tipo de comunicação entre os elementos, o middleware é classificado em síncrono ou assíncrono. Nos middleware que adotam o tipo de comunicação síncrona, os elementos permanecem bloqueados durante o tempo de comunicação. O elemento requisitante espera pela finalização do processo pelo elemento requisitado. Nos middleware assíncronos, o elemento requisitante é liberado logo após o envio do pedido de comunicação, e não permanece bloqueado à espera pela resposta da entidade requisitada.

Existem outras categorias de middleware que não foram discutidas nesta seção, tais como: reflexivos, adaptativos e cientes do contexto. Outros tipos de middleware foram desenvolvidos para os sistemas móveis, rede de sensores sem fio, tempo real e sistemas embarcados.

21.3 Serviços de middleware

As funcionalidades do middleware disponibilizadas para os programadores são denominadas serviços. Serviços são descritos por meio de uma interface e acessados através de uma API (Application Program Interface).

Não existe um único padrão para classificar os serviços de middleware. Em geral, os serviços são classificados em comuns e específicos. A Figura 21.2 ilustra um cenário no qual programadores de sistemas colaborativos interagem, por meio de uma API, com os serviços específicos e os comuns de um middleware.

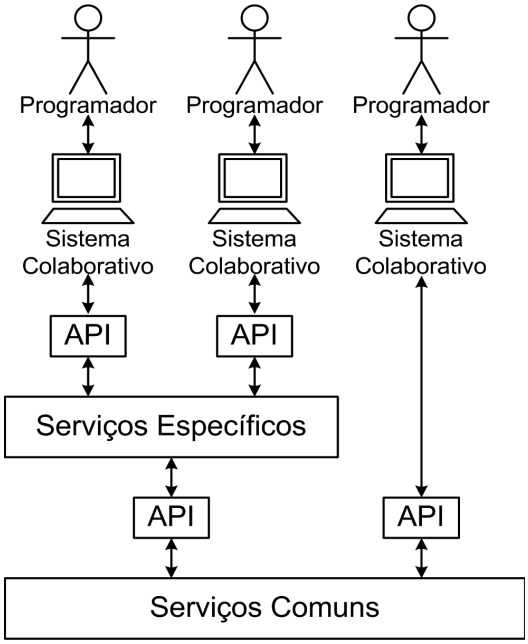


Figura 21.2 Serviços comuns e específicos em middleware

Os serviços comuns são de propósito geral e utilizados na implementação de sistemas de diferentes domínios. Estes serviços implementam funcionalidades relacionadas aos objetivos do middleware, tais como:

- Serviço de Nomes. Este é um serviço essencial para qualquer middleware, pois serve para dar transparência de localidade. O serviço de nomes possibilita o acesso a operações sem que o cliente precise conhecer a localização física (host, porta do processo e referência do objeto) do elemento que implementa cada operação. O endereço que o cliente precisa conhecer é o do servidor de nomes, que mantém os endereços dos outros serviços. Para facilitar a programação, um nome referencia um único objeto e está associado ao seu endereço físico. Assim como os nomes dos arquivos em relação aos diretórios, nomes de objetos devem ser únicos em um determinado contexto (naming context).

- Serviço de Eventos. Promove a comunicação entre objetos fornecedores de eventos e objetos consumidores por meio da notificação de ocorrência dos eventos de interesse. Este serviço implementa dois modelos: push e pull. No modelo push, produtores notificam os consumidores da ocorrência do evento, enquanto no modelo pull consumidores consultam os fornecedores sobre a ocorrência de eventos de interesse. Entre consumidores e fornecedores a comunicação é assíncrona.

- Serviço de Ciclo de Vida. Provê operações para implementar o ciclo de vida de um objeto: criação, destruição, movimentação e cópia. Objetos devem ser criados para fornecer ou solicitar funcionalidades, podem ser copiados ou movimentados de um local para outro, e destruídos quando deixam de ser usados.
- Serviço de Controle de Concorrência. Promove a coordenação do acesso concorrente de vários objetos a um determinado recurso compartilhado. O serviço deve garantir que o objeto servidor permaneça sempre em um estado consistente.
- Serviço de Comunicação. Possibilita a troca de mensagens entre elementos. Este serviço implementa as primitivas de interação do middleware: RPC, send, receive, entre outras.

Existem outros serviços comuns que um middleware pode disponibilizar como, por exemplo: persistência e transação. Serviços de persistência auxiliam na gerência da base de dados do sistema, serviços de transação podem implementar os protocolos para realização de uma transação na base de dados, como por exemplo, o protocolo "two phase commit".

Serviços específicos são direcionados para atender os requisitos recorrentes de sistemas de um determinado domínio, tais como: saúde, financeiro, governo, comércio eletrônico e, inclusive, para os domínios de sistemas tipicamente colaborativos como redes sociais, comunicação, entre outros. Enquanto os serviços comuns são implementados pelos desenvolvedores de middleware e compõem uma determinada plataforma (CORBA, J2EE, Java RMI, entre outras), serviços específicos são implementados pelos programadores dos sistemas distribuídos que usam middleware como plataforma de desenvolvimento.

Para apoiar o desenvolvimento de sistemas colaborativos, os serviços específicos podem usar os serviços comuns na sua composição. Por exemplo, para implementar um serviço para emissão de convites para uma reunião, o serviço específico "Emissão de Convites" usa os serviços comuns de nomes, eventos e comunicação. O serviço de eventos é usado para notificar os convidados, o serviço de comunicação para entregar as mensagens de notificação de forma assíncrona, e o serviço de nomes para prover a transparência da localização física de todos os serviços utilizados. Sendo assim, o desenvolvedor do serviço "Emissão de Convites" implementa apenas a seleção de pessoas a serem notificadas e a formatação do texto da mensagem do convite. O serviço específico de emissão de convites é composto por outros serviços comuns para formar uma unidade integrada.

COMO IMPLEMENTAR SERVIÇOS ESPECÍFICOS?

Serviços específicos podem ser implementados com o uso de diferentes tecnologias. Uma tecnologia comumente utilizada é a de Web Services que oferece um conjunto de padrões e mecanismos que apoiam funcionalidades frequentemente necessárias para o desenvolvimento de sistemas colaborativos distribuídos na web. Por exemplo, para a descrição de serviços é utilizada a linguagem WSDL (Web Service Description Language); para a localização de serviços, protocolos como UDDI (Universal Description, Discovery and Integration) podem ser usados; SOAP (Simple Object Access Protocol) para a troca de mensagens entre os serviços; aspectos relacionados à definição de um processo que envolve uma sequência de serviços podem ser apoiados pelo padrão WS-BPEL (Web Services - Business Process Execution Language). Outros padrões fazem parte da tecnologia de Web Services e podem apoiar diferentes aspectos na construção de serviços.

21.4 Middleware para apoiar a interoperabilidade

Plataformas de middleware tratam a interoperabilidade e a integração dos serviços. O suporte ao desenvolvimento de sistemas colaborativos vai além daquele necessário para o desenvolvimento de outros sistemas distribuídos. É preciso dar suporte para que diferentes grupos possam interagir mantendo a independência dos sistemas em relação à sua evolução. Por outro lado, sistemas que foram desenvolvidos em ambientes diversificados (cultura e tecnologia) talvez necessitem ser unificados numa única infraestrutura.

Para apoiar o desenvolvimento de sistemas colaborativos distribuídos, é comum observarmos duas soluções. A primeira é o uso de uma única infraestrutura (por exemplo, uma única plataforma de middleware), porém, esta solução não considera a diversidade de usuários, de requisitos e de contextos a ser tratada. A outra solução é o suporte a diferentes serviços como correio eletrônico, bate-papo e fóruns de discussão para apoiar as funcionalidades exigidas pelo sistema colaborativo. Esta segunda solução requer um esforço de programação para interoperar dados e procedimentos inerentes ao projeto de cada serviço utilizado.

A necessidade de apoiar interações entre grupos distribuídos demanda um suporte para tratar a diversidade de contextos nos quais os serviços que apoiam a coordenação e a comunicação estão inseridos. Serviços que apoiam a coordenação tratam as convenções distintas de acordo com a organização à qual eles estão apoiando. Serviços de comunicação implementam os protocolos compreendidos por cada um dos serviços envolvidos na troca de mensagens. Para que os serviços interoperem, um esforço de tradução é necessário para viabilizar a troca e compreensão de dados.

Frequentemente observamos que as atuais infraestruturas não tratam a complexidade relacionada à interoperabilidade entre sistemas colaborativos distintos. Soluções implementadas por essas infraestruturas exigem que os sistemas adotem representações comuns, por exemplo, nos modelos de dados. A adoção de uma plataforma de middleware para apoiar o desenvolvimento de sistemas colaborativos, portanto, deve evitar o esforço adicional tanto para o projeto quanto para a implementação das atividades distribuídas. Para que os sistemas se tornem interoperáveis é necessário que sejam disponibilizadas interfaces de acesso às operações dos serviços.

Middleware é um ponto de convergência para o processo de interoperabilidade. Se os sistemas usam um middleware, a interoperabilidade passa a ser de responsabilidade do middleware e não diretamente dos sistemas entre si. Imagine um cenário com seis sistemas distintos, sendo três implementados na plataforma "J2EE" e outros três na plataforma ".Net". Para que esses sistemas interoperem, é necessário incluir serviços para a troca de mensagens entre estes dois tipos de middleware. Se um middleware não estiver presente, é necessário desenvolver serviços de interoperabilidade responsáveis pelo entendimento e conversão dos dados trocados entre cada par de sistemas.

21.5 Middleware para apoiar a integração

Desenvolvedores de um sistema colaborativo, ao implementarem uma funcionalidade, necessitam interagir com outros sistemas e serviços. Neste contexto, dois cenários podem ser identificados: os outros sistemas já existem e complementam a funcionalidade do sistema que será

desenvolvido; ou então novos sistemas e serviços terão que ser desenvolvidos. Independente do cenário, os sistemas e serviços passarão a fazer parte de um sistema maior que necessita ser gerenciado. A Figura 21.3 ilustra o cenário de um sistema de edição cooperativa de textos. Consideremos, por exemplo, que no processo de apoio à edição seja necessário interagir com outras pessoas por meio de vídeo. Se o suporte para a interação for realizado por um sistema de videoconferência que não faça parte do editor de texto, as informações e decisões resultantes da interação serão persistidas fora do editor e, consequentemente, não são facilmente recuperadas e contextualizadas no documento editado. O histórico das decisões tomadas no processo de edição até poderá ser perdido. Organizações necessitam, portanto, convergir para a utilização de um conjunto de tecnologias que se complementem. As atividades de integração estão relacionadas aos problemas decorrentes da diversidade das tecnologias como protocolos, linguagens e plataformas de cada sistema.

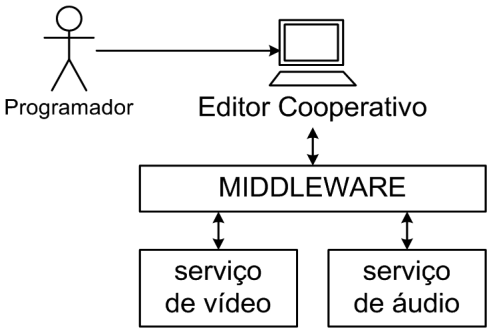


Figura 21.3 Middleware para integrar serviços em sistemas colaborativos

A integração de sistemas colaborativos se diferencia da integração de outros sistemas distribuídos. A complexidade da integração em sistemas colaborativos diz respeito aos diferentes modos e padrões de interação que deverão ser apoiados (síncrono e assíncrono), e, consequentemente, aos diferentes tipos e formatos de dados envolvidos. Por exemplo, em um editor cooperativo de texto, as atividades síncronas acontecem simultaneamente no texto compartilhado, o que requer serviços para a percepção das ações dos outros usuários. Já as atividades assíncronas, que acontecem em momentos distintos, requerem serviço para o versionamento, a persistência e a recuperação das modificações ocorridas entre os acessos. Para a integração é necessário um esforço considerável para unificar os dados, as tecnologias e protocolos usados entre os diferentes sistemas.

No contexto de suporte ao desenvolvimento de sistemas colaborativos distribuídos, a complexidade de integração está associada também às diferentes convenções adotadas pelos grupos, que variam de acordo com as suas constituições e organizações. Consideremos, por exemplo, dois editores de textos que necessitam ser integrados. Em um sistema existe o papel de coordenador da atividade, enquanto na outra não existe este papel. Nesse sistema, convencionou-se que os próprios participantes do processo de edição definiriam as formas de condução e o término da atividade. Integrar diferentes sistemas, portanto, significa correr o risco de aderência aos padrões previamente estabelecidos por cada sistema.

Os problemas de integração de sistemas colaborativos não estão restritos aos aspectos tecnológicos, mas também estão relacionados aos conceitos representados diferentemente por

cada sistema. Consideremos, por exemplo, os seguintes termos: "participante" em um fórum de discussão, e "usuário" em um editor de texto. Esses termos fazem referência ao mesmo conceito e precisam estar correlacionados na integração dos sistemas, caso contrário, os dados não serão trocados entre os diferentes sistemas. Adicionalmente, sistemas com diferentes modelos de dados, que representam os mesmos conceitos, necessitam ser correlacionados para que, posteriormente, possam ser integrados.

21.6 Benefícios e desafios do uso de middleware

Uma plataforma de middleware apoia o desenvolvimento de sistemas colaborativos distribuídos em diferentes níveis de abstração. Em relação aos aspectos tecnológicos, os serviços comuns podem oferecer o suporte adequado retirando do programador a necessidade de compreender detalhes para integrar os sistemas. Por outro lado, os serviços específicos podem ser utilizados para compor novos serviços e atender às especificidades de cada sistema. Por exemplo, no contexto de uma edição colaborativa, os serviços comuns estão relacionados aos aspectos de segurança e de concorrência para uma atividade de edição síncrona. Já os serviços específicos estão relacionados ao suporte às atividades inerentes aos sistemas colaborativos, tais como: controle de versão do documento, visualização das contribuições de cada participante, definição de papéis, apoio à comunicação entre os participantes, fornecimento de mecanismos de filtragem de informações, entre outras.

Quando os sistemas se tornam interoperáveis, deve-se manter a independência quanto à evolução e, portanto, é preciso disponibilizar as suas interfaces para outros sistemas. As atividades inerentes à interoperabilidade poderão ser realizadas por serviços específicos.

A partir do momento em que os serviços são integrados, os dados associados a cada serviço poderão fazer parte de um único banco de dados. Como resultado, um processo de entendimento do código que cada serviço implementa pode ser necessário. Um middleware representa, portanto, no contexto de desenvolvimento de sistemas colaborativos distribuídos, um ponto de convergência para os sistemas que necessitam ser integrados, diminuindo a complexidade desse processo.

A integração de serviços relacionados ao desenvolvimento de sistemas colaborativos também está relacionada à evolução dos processos de negócio das organizações. Muitas vezes, serviços são oferecidos no contexto de cada organização e a partir do momento em que os processos são unificados, esses serviços devem fazer parte de um cenário mais abrangente. Certamente, problemas serão evidenciados considerando-se, sobretudo, a ausência de padrões entre as organizações. Uma forma de amenizar esses problemas é estabelecer tecnologias comuns para os serviços, por exemplo: mesma linguagem de programação ou banco de dados. Porém, isso não solucionará esses problemas na sua totalidade. A utilização de uma infraestrutura de middleware apoia o processo de integração por oferecer serviços (específicos ou comuns) em diferentes níveis de abstração. Esses serviços podem implementar, por exemplo, procedimentos de conversão de protocolos ou de políticas para reduzir as complexidades inerentes de integração.

EXERCÍCIOS

21.1 Considere o desenvolvimento de um sistema colaborativo para apoiar atividades relacionadas à recomendação de filmes. Por meio da utilização desse sistema espera-se que os usuários, após se cadastrarem, informem os seus perfis e gostos por determinados estilos de filmes. Opiniões são associadas a cada filme, podendo dar início a uma discussão. Esse sistema deverá ser desenvolvido a partir da definição de dois cenários. São eles: (i) inicialmente, sistemas serão desenvolvidos para a execução em uma única organização; e (ii) sistemas em localidades distintas serão integrados. Para cada cenário, identifique e justifique os serviços específicos e comuns que poderão ser desenvolvidos.

21.2 Reflita sobre a importância para a adoção de uma infraestrutura de middleware para apoiar o desenvolvimento de sistemas colaborativos. Apresente três argumentos que justificam a adoção e, pelo menos, dois que podem impedir a sua utilização. Utilize exemplos para ilustrar a sua resposta.

21.3 Considere que um sistema para apoiar o processo de aprendizagem colaborativa necessita interoperar com um editor cooperativo de figuras através de uma infraestrutura de middleware. Além de apoiar a edição colaborativa de textos e figuras, o sistema possibilita que os usuários cadastrados recebam notificações sobre atividades ou convites para autoria de textos. Identifique as categorias de middleware (MOM, MP, MOO, MT) que a plataforma deve possuir para que o programador desenvolva os serviços do sistema. Justifique sua escolha.

21.4 Em relação aos aspectos de transparência que um middleware oferece para o desenvolvimento de sistemas colaborativos distribuídos, pesquise outros tipos de transparência além dos citados no corpo deste texto.

21.5 Uma organização, com departamentos geograficamente distribuídos, possui um sistema colaborativo para apoiar reuniões. Os funcionários dessa organização podem ser eventualmente convidados para uma reunião virtual. Neste sistema ainda não foi disponibilizada uma lista de usuários "online", de tal forma que, por meio desta lista, uma comunicação instantânea possa ser estabelecida entre eles.

Considere que serão desenvolvidos dois serviços específicos para este sistema: um para apoiar as atividades de coordenação e outro para apoiar a comunicação entre os participantes da reunião. Especifique pelo menos dois requisitos funcionais para cada um desses serviços e a forma pela qual eles podem interoperar.

LEITURAS RECOMENDADAS

- IT Architectures and Middleware – Strategies for Building Large, Integrated Systems (Britton e Bye, 2009). Neste livro são apresentados os conceitos envolvidos na tecnologia de middleware, os elementos e os aspectos que justificam a adoção de middleware. Também são discutidos os princípios de sistemas distribuídos, bem como o suporte que a tecnologia de middleware oferece para a integração de sistemas.
- Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications (Schantz e Schmidt, 2001). Neste artigo são apresentados os principais

elementos de middleware orientados a objetos. Adicionalmente, é apresentada uma taxonomia para classificar os serviços deste tipo de middleware. Esta taxonomia foi usada em nosso capítulo.

- Distributed Systems Architecture: A Middleware Approach (Pudder, Römer e Pilhofer, 2006). Os conceitos básicos sobre middleware estão nesta obra. Alguns tipos de middleware são exemplificados, bem como o processo de desenvolvimento de sistemas que se beneficiam do suporte oferecido por estas infraestruturas. Mais ainda, aborda tecnologias, como CORBA e Web Services, e explora conceitos de interoperabilidade e integração.

REFERÊNCIAS

BRITTON, C., BYE, P., IT Architectures and Middleware – Strategies for Building Large, Integrated Systems, Addison-Wesley, Second Edition, 2004.

SCHANTZ, R., SCHMIDT, D., Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications, Encyclopedia of Software Engineering, Wiley & Sons, 2001.

PUDDER, A., RÖMER, K., PILHOFER, F., Distributed Systems Architecture: A Middleware Approach, Morgan e Kauffman Publishers & Elsevier, 2005.