CAPÍTULO 22

Componentes de software para sistemas colaborativos

Marco Aurélio Gerosa
Igor Steinmacher

META

Apresentar os conceitos da tecnologia de componentes de software para o desenvolvimento de sistemas colaborativos.

OBJETIVOS EDUCACIONAIS

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

- Listar benefícios e desafios relacionados ao uso de componentes para o desenvolvimento de sistemas colaborativos.
- Aplicar os conceitos de desenvolvimento baseado em componentes para análise e construção de sistemas colaborativos.

RESUMO

A construção de sistemas colaborativos envolve dificuldades técnicas e multidisciplinares. A tecnologia de componentes reduz essas dificuldades, favorecendo a prototipação e a experimentação, o desenvolvimento iterativo e a adaptação dos sistemas para diversas situações. A tecnologia de componentes já vem sendo usada com sucesso em vários domínios de aplicação. Neste capítulo são discutidos os conceitos, as vantagens e os desafios do Desenvolvimento Baseado em Componentes (DBC) para o desenvolvimento de sistemas colaborativos. São discutidas a Engenharia de Domínio, que se preocupa com a identificação de características e desenvolvimento de artefatos para reúso; e a Engenharia de Aplicação, que define as atividades necessárias para desenvolver aplicações com base no reúso de artefatos e de modelos. São apresentados, também, alguns kits para a construção de sistemas colaborativos com componentes.

22.1 Desenvolvimento baseado em componentes

Você conhece o brinquedo LEGO tão comum entre as crianças? São peças com diferentes tamanhos, formatos, cores e funções, e todas seguem um padrão de encaixe que possibilita sua combinação para construir diferentes produtos. No contexto de desenvolvimento de software, componentes têm um significado semelhante: são "peças" de software com funções específicas, combináveis e reusáveis. O que define um componente de software é justamente a padronização, que inclui o encaixe, o encapsulamento, a implantação, a infraestrutura de execução entre outras características.

ORIGEM DA COMPONENTIZAÇÃO NA ENGENHARIA DE SOFTWARE

A ideia de componentização é tão antiga quanto o termo Engenharia de Software. Na mesma conferência em que esse termo foi definido, em 1968, McIlroy introduziu o conceito de componentes e reúso em seu artigo "Mass Produced Software Components" (McIlroy, 1968). Nos anos 1990, o Desenvolvimento Baseado em Componentes tornou-se popular e passou a ser estudado com mais profundidade na academia. A tecnologia foi popularizada principalmente pelos componentes de interface gráfica com o usuário (widgets) usados para compor visualmente uma aplicação. O uso dessa tecnologia simplificou consideravelmente o desenvolvimento de aplicações desktop, tornando possível construir aplicação apenas arrastando instâncias dos componentes para uma tela, configurando suas propriedades e programando o comportamento das instâncias. Com o advento da web, componentes encapsulados como serviços vêm sendo utilizados para composição e combinação de funcionalidades providas por diferentes fontes. Essas composições vêm sendo chamadas de mashups.

22.2 Sistema colaborativo baseado em componentes

O desenvolvimento de sistemas colaborativos tem alta complexidade técnica e requer conhecimento multidisciplinar. Envolve áreas de estudo além da computação, como sociologia, psicologia, antropologia e administração. As questões técnicas são complexas porque um sistema colaborativo é multiusuário e geralmente distribuído, o que demanda conhecimentos sobre conexões, protocolos, compartilhamento de recursos, concorrência de acesso, distribuição, gerenciamento de sessões, multiplicidade de plataformas etc.

O uso de componentes de software possibilita encapsular complexidades técnicas e multidisciplinares, e o desenvolvedor não precisa conhecer os detalhes do funcionamento interno dos componentes para montar os sistemas. O baixo grau de acoplamento entre componentes e as interfaces bem definidas favorecem o desenvolvimento distribuído e a alocação de diferentes desenvolvedores para partes específicas do sistema, o que é desejável porque dificilmente uma única pessoa detém o conhecimento para desenvolver todo um sistema colaborativo. Geralmente o desenvolvimento é realizado por equipes multidisciplinares.

Por envolver diferentes usuários, os requisitos de um sistema colaborativo raramente são claros o suficiente para uma especificação precisa e completa no início do desenvolvimento. O uso de componentes favorece a prototipação e o desenvolvimento iterativo. Os requisitos são levantados continuamente e clarificados a cada iteração, e o sistema é recomposto e reconfigurado. A componentização facilita experimentar diferentes formas de interação e ajustar o sistema às características do grupo e das tarefas.

A complexidade técnica para o desenvolvimento de sistemas colaborativos frequentemente se torna o foco da equipe de desenvolvimento. Ao isolar os detalhes de baixo nível por meio da tecnologia de componentes, os desenvolvedores conseguem dar mais atenção para os problemas específicos de colaboração. O uso de componentes facilita a prototipação, a experimentação e o feedback rápido, o que leva à melhoria contínua e favorece a criatividade e a exploração na busca de boas soluções para o suporte à colaboração. A componentização possibilita replicar e evoluir as soluções, o que potencialmente promove a definição de boas práticas, guias e padrões para o desenvolvimento de sistemas colaborativos. Conforme aumentam as dificuldades técnicas, também aumentam as vantagens do uso de componentes para o encapsulamento das complexidades.

Sistemas colaborativos são usados em larga escala pela web. A ênfase na interação e colaboração promoveu o conceito de Web 2.0, termo usado para caracterizar o suporte à interação social, inteligência coletiva, interoperabilidade, múltiplas formas de acesso e grandes volumes de dados. Os sistemas Web 2.0 estão sempre em desenvolvimento, em um estado denominado beta perpétuo. Atualizações e evoluções ocorrem regularmente, diluindo o conceito de versão e lançamento. Em uma abordagem baseada em componentes, os sistemas evoluem pela atualização de componentes já existentes e pela adição de novos componentes sem a necessidade de gerar novas versões uma vez que componentes são trocados e adicionados a qualquer momento no sistema sem reinstalações.

22.3 Engenharia de software baseada em componentes

Reusar componentes prontos e testados em múltiplos sistemas e contextos favorece a redução do tempo de desenvolvimento e o aumento da qualidade do sistema. A modularização

propiciada pela tecnologia de componentes também favorece a manutenção dos sistemas e o desenvolvimento em paralelo. Também se destacam as vantagens: melhor suporte à prototipação e evolução do software, melhor capacidade de adaptação, substituição dinâmica de partes do sistema, gerenciamento de mudanças, interoperabilidade, entre outras.

O desenvolvimento baseado em componentes, em contrapartida, também apresenta algumas dificuldades: dificuldades do desenvolvimento de componentes e dificuldades do desenvolvimento com componentes.

No desenvolvimento de componentes, torna-se necessário um maior esforço inicial de análise, projeto e implementação para montar a infraestrutura do sistema e construir uma biblioteca robusta de componentes reusáveis. Projetar e preparar um pedaço de software para futuro reúso aumenta a necessidade de flexibilidade, documentação, estabilidade e abrangência do software. A literatura estima que o custo inicial de desenvolvimento dos componentes e da infraestrutura de execução é pago somente a partir do 3º reúso.

No desenvolvimento com componentes, caso os componentes sejam de terceiros, devem ser consideradas: a curva de aprendizagem, a necessidade de adaptar os requisitos aos componentes e a adaptação do processo de desenvolvimento. A curva de aprendizagem é relacionada ao estudo necessário para instalar e usar os componentes. Às vezes é mais rápido desenvolver um componente do que procurar por um pronto, estudá-lo e adaptá-lo. A menos que o custo de aprendizagem seja amortizado por vários projetos ou que o ganho de produtividade e qualidade seja expressivo, o investimento inicial não é atraente. O reúso de componentes provenientes de terceiros eventualmente leva a situações inesperadas. Há o risco de incorporar erros de software produzidos por terceiros. Muitas vezes é difícil encontrar um componente que atenda plenamente às funcionalidades desejadas, o que torna necessário implementar uma grande quantidade de código para alterar ou customizar os componentes. Os componentes também introduzem dependências fora do controle dos desenvolvedores do sistema, impondo um esforço contínuo de atualização das versões e reconfiguração. Em função dessas dificuldades, os componentes COTS (Commercial Off-The-Shelf), que são componentes genéricos prontos para uso comprados ou licenciados de terceiros, têm um nicho de aplicação bem mais restrito do que se acreditava até o início da década de 1980, quando se almejava a criação de repositórios globais de componentes de propósito geral que possibilitariam montar a maioria dos sistemas, similar ao que acontece na eletrônica. Hoje, a tecnologia de componentes é usada em contextos específicos e bem definidos.

Dentro da disciplina de Engenharia de Software, encontra-se uma especialização voltada para componentes: a Engenharia de Software Baseada em Componentes (CBSE – Component Based Software Engineering). Engloba técnicas e tecnologias para desenvolver componentes reusáveis, adaptáveis e atualizáveis, e também para a utilização de componentes na composição de sistemas maiores.

22.3.1 Engenharia de Domínio: desenvolvimento de componentes

Na Engenharia de Domínio busca-se identificar características comuns de uma família de sistemas para a qual são desenvolvidos e disponibilizados componentes reusáveis. A Engenharia de Domínio contempla as seguintes atividades: Análise de Domínio, Desenvolvimento de uma Arquitetura de Domínio e o Desenvolvimento de Componentes de acordo com a arquitetura definida – Figura 22.1.

Na Análise de Domínio, objetiva-se identificar e classificar as semelhanças e variações entre os sistemas de um domínio de aplicação. A identificação das características comuns facilita o reúso e a comunicação, pois fornece um vocabulário compartilhado entre os desenvolvedores e os usuários. Por exemplo, sobre "redes sociais", são identificadas várias características comuns entre os sistemas desse domínio: definição de perfil, busca de pessoas, mural de mensagens, comentários, compartilhamento de imagens etc.

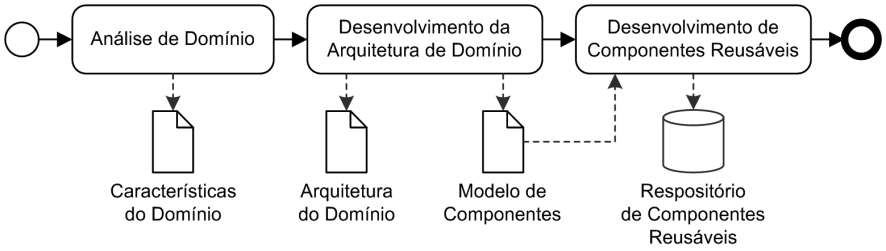


Figura 22.1 Engenharia de Domínio Baseada em Componentes

O Desenvolvimento da Arquitetura do Domínio envolve a especificação e a criação de uma arquitetura de software a partir de requisitos especificados a partir das características identificadas na Análise de Domínio. As características são mapeadas para soluções técnicas e usadas durante o desenvolvimento do sistema. Como resultado dessa atividade, obtém-se a definição de um modelo de componentes (ver Seção 22.4).

Na atividade de Desenvolvimento de Componentes Reusáveis, são construídos os componentes de software identificados e classificados durante a Análise do Domínio seguindo o modelo de componente especificado na atividade de Desenvolvimento da Arquitetura. Depois de construídos, os componentes são armazenados em um repositório de componentes para ficarem disponíveis para o desenvolvimento de futuros sistemas.

22.3.2 Engenharia de Aplicação: desenvolvimento com componentes

A Engenharia de Aplicação tem por objetivo o desenvolvimento de sistemas por meio da composição de componentes. As atividades da engenharia de aplicação são apresentadas na Figura 22.2.

Para a Análise e o Projeto da Aplicação, são usados métodos tradicionais da Engenharia de Software, porém, são considerados os modelos gerados nas atividades de Análise e Arquitetura da Engenharia de Domínio. A atividade seguinte é a Seleção de Componentes. São reusados e adaptados os componentes encontrados que estejam adequados ao novo sistema, e também são construídos novos componentes de acordo com as práticas da Engenharia de Componentes. É verificado se os componentes selecionados no repositório atendem às funcionalidades desejadas e se encaixam no sistema em desenvolvimento. Deve-se considerar a cobertura dos requisitos pelos componentes selecionados, as bibliotecas e aplicações exigidas pelos componentes, a integração com outros componentes já existentes no sistema e a compatibilidade com recursos de hardware e sistema operacional previstos. Depois de selecionados, os componentes são ajustados em função da arquitetura, são integrados com outros componentes, e eventualmente são adaptados em função de alguma característica que ainda

não esteja de acordo com os requisitos. Essa atividade é chamada de Adaptação dos Componentes. Após as adequações, os componentes são submetidos a Testes de Componentes para verificar a conformidade com as funcionalidades desejadas.

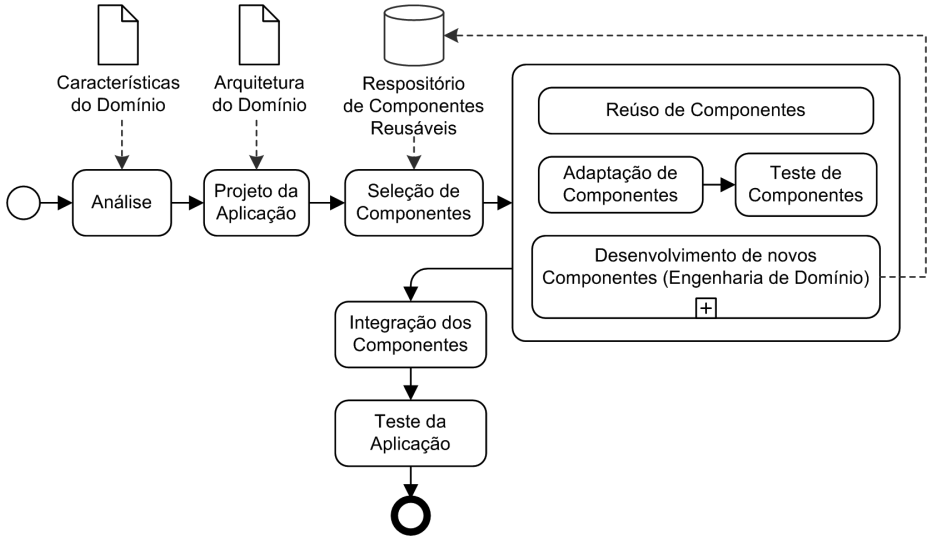


Figura 22.2 Engenharia de Aplicação baseada em componentes

Depois de selecionados, é preciso integrar os componentes para compor o sistema. Para a atividade de Integração dos Componentes, é utilizada uma infraestrutura de acordo com a arquitetura definida, o que inclui intercâmbio e armazenamento de dados e mecanismos de acoplamento de componentes. Na integração é feita a implantação dos componentes na infraestrutura de execução. Por fim, no Teste de Aplicação o sistema é testado com todos os componentes combinados.

Para exemplificar a engenharia de aplicação, voltemos a discutir o nosso exemplo de sistema de redes sociais. São buscados componentes no repositório que dão suporte às características desejadas para o nosso sistema, e são selecionados os mais adequados aos requisitos e à arquitetura (Seleção dos Componentes). Alguns ajustes eventualmente são feitos para que os componentes selecionados fiquem de acordo com a arquitetura e com o leiaute do sistema (Adaptação de Componentes). Caso em nosso sistema esteja prevista a troca de arquivos pelo bate-papo, e não encontremos este serviço no repositório, então precisaremos construir um novo componente (Engenharia de Componentes). Combinamos os componentes com a infraestrutura para a criação do nosso novo sistema de redes sociais (Integração dos Componentes). Por fim, após a combinação de todos os componentes, realizamos testes sobre o sistema desenvolvido (Teste de Aplicação).

22.4 Modelo de componentes

O termo componente é usado na literatura com pouco rigor, muitas vezes como sinônimo de elemento ou módulo de software. Neste capítulo, definimos componente de software de forma específica: pedaço de software executável, com o escopo de funcionamento bem definido, e que esteja em conformidade com um modelo de componentes.

Para ilustrar a definição de componentes, considere os plugins dos navegadores web usados para estender o suporte à visualização de conteúdos, como animações em flash e documentos em PDF. Esses plugins são componentes de software, uma vez que proveem serviços específicos de maneira bem definida, seguem um modelo padrão de componentes e são disponibilizados como uma unidade executável em uma plataforma predefinida.

O conceito de componente é similar ao conceito de módulo no sentido que ambos dividem um produto em partes menores, coesas e fracamente acopladas. Entretanto, diferentemente dos módulos, os componentes seguem uma série de padrões, possuem uma infraestrutura específica para o gerenciamento e a execução, e são encapsulados em uma unidade executável.

Também há uma confusão entre os conceitos de componentes e objetos. Ambos possuem características semelhantes, como a necessidade de apresentarem interfaces bem definidas para disponibilizar as operações. Porém, componentes são desenvolvidos em qualquer linguagem de programação. Objetos são instâncias de classes que até podem ser usados para implementar componentes. Os componentes, em contrapartida, são combinados para a criação de sistemas em tempo de implantação.

Um componente normalmente é passível de ser implementado e mantido independentemente de um sistema. Deve ser coeso, não trivial, e com uma função clara no contexto da arquitetura. Também são características desejáveis dos componentes: reúso, substituição e combinação com outros componentes.

TIPOS DE COMPONENTES

Componentes diferem em complexidade, escopo, grau de funcionalidade, habilidades necessárias para usá-los e infraestrutura necessária. São classificados em três tipos:

- Componentes de Interface com o Usuário. Essa categoria de componentes inclui botões, campos de texto, ícones, janelas e listas. Esses componentes são comumente chamados de widgets ou gadgets. É o tipo mais comum encontrado de componente, devido à simplicidade de construção e baixo grau de complexidade.
- Componentes de Serviço. Esses componentes proveem serviços que são comuns a diversos tipos de sistemas, como acesso a banco de dados, acesso a serviços de mensagens, transações e integração de sistemas. O custo de desenvolvimento desse tipo de componente é mais alto se comparado aos componentes de interface com o usuário.
- Componentes de Domínio. São os componentes mais difíceis de desenvolver e reusar. Exemplos desse tipo incluem componente de reunião virtual e de edição cooperativa de documentos. Esses componentes são desenvolvidos para um domínio específico, são mais difíceis de serem projetados e construídos, pois requerem um alto grau de expertise e maior flexibilidade para adequá-los a diversas situações de uso.

Componentes de software precisam estar em conformidade com um modelo de componentes. Um modelo de componentes estabelece padrões de implementação e documentação para possibilitar a interação e a composição dos componentes. Alguns exemplos de modelos de componentes são: Corba Component Model (CCM), Microsoft OLE, (D)COM/COM+, Enterprise Java Beans e Web Services.

Para que os componentes se "encaixem", é preciso seguir um padrão de interface definido por um modelo de componentes. A interface serve como um contrato que especifica os serviços prestados. Cabe ao componente a responsabilidade de implementar os serviços. Alguns modelos de componentes usam uma linguagem de descrição de interface (IDL) independente da linguagem de programação. Outros usam a própria linguagem ou tecnologia para definir a interface, como é o caso das interfaces em orientação a objetos.

Componentes devem ter nomes diferentes para evitar conflitos, o que é definido pelo padrão de nomenclatura. Alguns modelos definem identificadores globais únicos gerados pela combinação de alguns dados, como a data de criação do componente e o endereço da placa de rede, como é o caso do modelo COM (Component Model). Outros modelos definem um espaço hierárquico de nomes, como é o caso da tecnologia Java que utiliza o endereço de domínio da instituição invertido.

Componentes precisam estar interconectados e trocar informações com componentes adquiridos de diferentes fornecedores. Um modelo de componentes define padrões de interoperabilidade para a comunicação entre componentes escritos em diferentes linguagens ou implantados em diferentes máquinas. Dentre os padrões mais usados no mercado: SOAP (Simple Object Access Protocol), que usa um padrão baseado em XML para troca de mensagens independente da linguagem de programação; e RMI (Remote Method Invocation), que executa chamadas remotas em aplicações desenvolvidas na plataforma Java. Caso os componentes não sejam interoperáveis, é necessário construir um adaptador ou "código cola" entre os componentes.

Um modelo de componentes também define padrões de composição. Os tipos de acoplamento mais comuns entre componentes são: cliente-servidor, em que o cliente explicitamente chama operações do servidor; e publicador-ouvinte, em que o ouvinte se registra como tratador de eventos e de informações disponibilizadas pelo publicador.

Um modelo de componentes também define regras e padrões de evolução dos componentes, incluindo o versionamento. Também define um padrão de instalação que especifica o formato dos pacotes e o processo de registro e ativação dos componentes na infraestrutura.

22.5 Toolkits e plataformas para construção de sistemas colaborativos com componentes

Nesta seção são apresentados alguns pacotes que implementam componentes específicos e oferecem infraestrutura para a construção de sistemas colaborativos. Por apresentar um kit para "montagem" de aplicações, são chamados de toolkits.

GroupKit é um dos primeiros toolkits de componentes especializados para a construção de sistemas colaborativos. Contêm componentes e uma plataforma de execução para o desenvolvimento de sistemas colaborativos síncronos. Oferece facilidades aos programadores para

interconexão, gerenciamento de eventos, comunicação entre componentes e compartilhamento de dados. O GroupKit oferece diversos widgets colaborativos para compor a interface gráfica da aplicação, como: teleapontadores, barras de rolagem multiusuário, visão de radar da área compartilhada etc. Esses widgets são particularmente relevantes para a construção de interfaces WYSIWIS relaxadas, onde as dimensões das janelas dos usuários podem ser distintas. SDGToolkit (kit para Single Display Groupware) é um kit de componentes derivado do GroupKit que tem por objetivo a criação de sistemas para apoiar o trabalho de grupos presenciais usando um monitor compartilhado. O SDGToolkit gerencia automaticamente múltiplos mouses e teclados e apresenta múltiplos cursores na tela. Provê aos desenvolvedores mecanismos para captura e manipulação de eventos dos dispositivos de entrada, e mecanismos para o desenvolvimento de widgets para novos sistemas. Esse mesmo grupo responsável pelo GroupKit e SDGToolkit também disponibilizou widgets que encapsulam detalhes técnicos da interação com dispositivos físicos, denominados phidgets <http://www.phidgets.com>. Dispositivos de hardware de entrada e saída – como botões, acelerômetros, sensores e servomotores – são encapsulados pelos phidgets de tal forma que os programadores criam sistemas integrados a dispositivos físicos de maneira similar à construção de interfaces gráficas com widgets. A implementação e os detalhes de construção são escondidos pela exposição da funcionalidade do dispositivo por meio de uma API.

GROUPKIT E GROUPLAB

GroupKit (Roseman e Greenberg, 1997) <http://www.groupkit.org> foi criado em 1992 por Mark Roseman como parte do trabalho de graduação desenvolvido no Projeto GroupLab do famoso pesquisador Saul Greenberg. No site do GroupLab <http://grouplab.cpsc.ucalgary.ca> estão disponíveis vários conteúdos interessantes sobre sistemas colaborativos, incluindo textos, vídeos e vários toolkits que estão disponíveis na seção de software. Vale a visita.

Componentes de software para colaboração também estão disponíveis nos gerenciadores de conteúdo (CMS – Content Management System). Esses sistemas apresentam uma arquitetura modular para a construção de portais que oferecem suporte à interação com os usuários. Vários componentes dão suporte à colaboração, como bate-papo, fóruns, sistemas de votação e reputação, compartilhamento de imagens, comentários, wiki etc. São vários os exemplos desse tipo de sistema: Joomla, Mambo, Xoops, Zope/Plone, WordPress e Drupal.

Sobre redes sociais, há iniciativas voltadas para a componentização, como o projeto OpenSocial <http://www.opensocial.org> que define um modelo contendo APIs para o desenvolvimento de componentes e acesso a dados de redes sociais existentes. Esse modelo propicia o desenvolvimento de widgets que funcionam em qualquer aplicação web compatível com o OpenSocial. Outra iniciativa é o projeto SocialSite <https://socialsite.dev.java.net>, que define uma infraestrutura para facilitar a inserção de funcionalidades de redes sociais em páginas web. A infraestrutura do SocialSite provê uma API para o armazenamento de grafos de redes sociais e um conjunto de widgets e serviços para acesso e manipulação das redes sociais, tais como: perfil de usuário, log de atividades, grupo de amigos e políticas de privacidade.

KITS DE COMPONENTES VERSUS FRAMEWORKS

Frameworks também são desenvolvidos para o reúso. Um framework é uma infraestrutura reusável de todo ou de parte de um sistema, para que seja instanciado para desenvolver uma família de sistemas. As partes invariantes de um domínio são implementadas no framework e reusadas nas instanciações. Por definir uma arquitetura parcialmente implementada e encapsular detalhes de implementação, o framework também libera os desenvolvedores das complexidades técnicas envolvidas na solução. O framework geralmente também é construído por especialistas de um domínio e reusado por leigos naquele domínio.

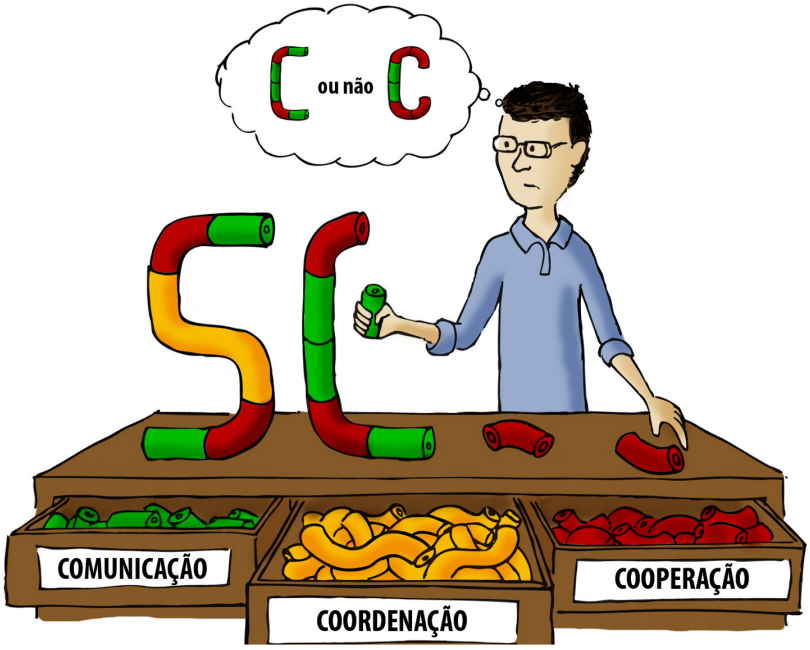
A principal diferença entre um framework e um kit de componentes é a forma como o sistema é instanciado. Um framework tem uma infraestrutura fixa e um fluxo de chamadas de operações que limita as extensões aos pontos previamente concebidos, denominados hot spots. Em uma arquitetura baseada em componentes, estes podem ser usados em configurações não previstas inicialmente pelos desenvolvedores. Pelo baixo acoplamento entre os componentes, costuma ser mais fácil atualizar ou substituir um componente do que alterar uma parte de um framework. O framework possibilita um desenvolvimento mais imediato e menos propenso a erros, pois a forma de reusá-lo foi prevista e as opções são mais limitadas. Por outro lado, componentes oferecem mais flexibilidade e são manipulados em tempo de implantação, configuração e até mesmo em tempo de execução. Vale ressaltar que um framework pode ser construído a partir de componentes, e um componente pode ser construído a partir de um framework.

22.6 Desafios da tecnologia de componentes

A motivação inicial do desenvolvimento baseado em componentes era a criação de um mercado global de disponibilização, venda e reúso de componentes genéricos, de modo que a construção de software se tornasse uma atividade de composição a partir de componentes pré-elaborados por terceiros. A ideia que tanto motivou a comunidade científica no princípio mostrou-se inviável. Na prática, raramente são encontrados componentes genéricos que atendam perfeitamente às necessidades de negócio, que acabam sendo muito específicas. Componentes que foram construídos para serem altamente flexíveis tornam-se difíceis de usar e configurar. Muitas vezes o tempo para buscar, selecionar, estudar e adaptar componentes supera o de desenvolver código novo, de modo que empresas e desenvolvedores acabam optando por não correr o risco. Com isso, componentes passaram a ser usados em nichos específicos e com escopo mais controlado.

Várias empresas passaram a modularizar seus sistemas na forma de componentes para promover o reúso interno, o aumento da qualidade e a redução de custos. Uma técnica que vem sendo frequentemente empregada é a Linha de Produto de Software (LPS), que consiste no desenvolvimento de um conjunto de aplicações de um mesmo domínio construídas sobre uma arquitetura comum. A cada vez que uma nova aplicação é necessária, um conjunto de componentes específicos é selecionado a partir das características do domínio.

Um fator que motiva o uso de componentes é a possibilidade de composição de aplicações pelo usuário final (tailorability), como exemplificam os plugins de navegadores e de ambientes de desenvolvimento de software. Outro motivo é a utilização de serviços dinâmicos, em que



GROUPWARE WORKBENCH: COMPONENTES 3C PARA CRIAÇÃO DE SISTEMAS COLABORATIVOS

Groupware Workbench <http://www.groupwareworkbench.org.br> é desenvolvido e mantido pelo grupo no qual fazem parte os autores deste capítulo. Contém componentes de domínio classificados em função do Modelo 3C de Colaboração para a construção de sistemas colaborativos, e contém uma infraestrutura de execução para a Web 2.0. Os componentes do Groupware Workbench levam em conta o aspecto de mobilidade previsto na Web 2.0 e são integrados à plataforma Android, sistema operacional da Google para celulares. O Groupware Workbench é disponibilizado como software livre sob licença LGPL 3.0.

Groupware Workbench é fornecido em duas partes: o núcleo e os kits de componentes. O núcleo da bancada oferece suporte à instalação, atualização, agrupamento, customização, disponibilização, reúso, interdependências e ciclo de vida dos componentes, que são chamados de "collablets". Os componentes do Groupware Workbench encapsulam o suporte a diversas características recorrentes em aplicações Web 2.0, como por exemplo, gerenciamento de comentários, tags, usuários, papéis, perfis, conteúdos, georreferenciamento, recomendação, categorização, entre outros.

Groupware Workbench foi projetado para facilitar a construção de sistemas colaborativos, inclusive pensando no desenvolvimento por alunos de graduação em computação. Também é usado como plataforma para a construção de sistemas profissionais, como exemplifica o sistema de rede social para compartilhamento de imagens na área de arquitetura <http://www.arquigrafia.org.br>.

uma aplicação consulta catálogos de serviços e instala os que satisfazem as necessidades. Um exemplo de serviço conhecido é a busca de CEPs disponibilizada pelos correios e usada em diversos sites para facilitar o preenchimento de endereços.

Ainda há desafios a serem vencidos com relação ao desenvolvimento baseado em componentes, tais como: falta de confiança nos componentes, ausência de garantia de combinação e coexistência com outros componentes, e política de atualização e suporte. Desenvolvedores frequentemente perguntam: quem me garante que o componente é bom o suficiente? Será que quando eu 'encaixar' o componente em meu sistema o restante continuará funcionando? Quem é o responsável e como se dará a atualização desses componentes? Quem dará o suporte aos componentes? Também ocorre a dificuldade para definir a propriedade intelectual: quem é o "dono" de um componente ou de uma ideia adicionada a um componente?

Os desafios precisam ser discutidos, analisados e superados, para que a tecnologia de componentes possa ser cada vez mais aplicada como estratégia para o reúso do software, reduzindo a complexidade e, por consequência, melhorando o desenvolvimento dos sistemas colaborativos.

EXERCÍCIOS

22.1 Tendo em vista os benefícios da adoção da tecnologia de componentes para a construção de sistemas colaborativos, elabore uma lista ordenada com as cinco principais vantagens dessa abordagem em sua opinião. Discuta com um grupo as diferentes ordenações, e gere uma ordenação do grupo. Depois compare as diferentes ordenações geradas na turma.

22.4 Você é o engenheiro de domínio de uma empresa que deseja atuar na área de redes sociais. A empresa solicitou que você faça uma análise preliminar de domínio considerando os principais concorrentes de mercado (Facebook, Orkut, Ning etc.). Conforme exemplifica a tabela apresentada a seguir, identifique as características presentes nos sistemas analisados e classifique-as como desejável ou essencial.

	ESSENCIAL (E) / DESEJÁVEL (D)	SISTEMA 1	SISTEMA 2	SISTEMA 3
EXEMPLO: PERFIL DO PARTICIPANTE	*E*	✔	✔	✔
CARACTERÍSTICA 1				
CARACTERÍSTICA 2				
CARACTERÍSTICA 3				
...				

22.2 No texto são apresentados alguns desafios na área de desenvolvimento baseado em componentes. Liste três desafios e, para cada um, argumente sobre possíveis soluções que poderiam ser adotadas para resolvê-los.

22.3 Descreva três "componentes de domínio" e três componentes de "interface com o usuário" que sejam voltados para a construção de sistemas colaborativos.

22.5 Faça uma pesquisa sobre kits de componentes usados na construção de sistemas colaborativos. Um ponto de partida são os kits apresentados no capítulo.

LEITURAS RECOMENDADAS

- Toolkits and interface creativity (Greenberg, S., 2007). Nesse artigo é abordada a relação entre componentes de software e o aumento da criatividade na construção de sistemas colaborativos. O autor discute como o uso de toolkits traz avanços rápidos, e são apresentados casos de sucesso.
- CRUISE – Component-Reuse In Software Engineering (Almeida, E.S. et al., 2007). Esse livro contém uma revisão e análise aprofundada sobre reúso de software, e o estado da arte nessa área. O livro está disponível para download <http://cruise.cesar.org.br>
- Component-Based Software Engineering: Putting the Pieces Together (Heineman, G.T. e Councill W.T., 2001). Nesse livro você encontrará conceitos mais detalhados sobre o Desenvolvimento Baseado em Componentes: definições, processos de Engenharia de Software baseada em componentes, e construção e gerenciamento de sistemas baseados em componentes.
- Component Software – Beyond Object-Oriented Programming (Szyperski, C., 2002). Esse livro oferece uma análise da tecnologia de componentes que facilita a compreensão de como desenvolver orientado a componentes.
- Component technology: what, where, and how? (Szyperski, C., 2003). Artigo que apresenta motivações para usar componentes, onde e como são usados, e discute alguns problemas dessa abordagem.

REFERÊNCIAS

ALMEIDA, E.S., ALVARO, A., GARCIA, V.C., MASCENA, J.C.C.P., BURÉGIO, V.A., NASCIMENTO, L.M., LUCRÉDIO, D., MEIRA, S.R. CRUISE – Component-Reuse In Software Engineering, C.E.S.A.R e-book, 2007.

HEINEMAN, G.T., COUNCILL, W.T. Definition of a Software Component and Its Elements, in: Component-Based Software Engineering: Putting the Pieces Together, Addison-Wesley Professional, 2001.

GREENBERG, S. Toolkits and Interface Creativity, Journal of Multimedia Tools and Applications, v.32 (2), Special Issue on Groupware and Multimedia, Kluwer, pp. 139-159, 2007.

MCILROY, M.D. Mass-Produced Software Components, Software Engineering: Report on a Conference by the NATO Science Committee, P. Naur and B. Randell, eds., NATO Scientific Affairs Division, Brussels, pp. 138-155, 1968.

O'REILLY, T. What is Web 2.0: design patterns and business models for the next generation of software, 2005. Disponível em: http://www.oreilly.com/go/web2.

PRESSMAN, R.S. Engenharia de Software, 6 ed., São Paulo, McGrawHill, 2006.

ROSEMAN, M.; GREENBERG, S. Building real time groupware with GroupKit, a groupware toolkit. ACM Transactions on Computer-Human Interaction, v.3(1), p. 66-106, 1997.

SZYPERSKI, C. Component Software – Beyond Object-Oriented Programming, 2 ed., Addison-Wesley Professional, 2002

SZYPERSKI, C. Component technology: what, where, and how?. In Proceedings of the 25th international Conference on Software Engineering. IEEE Computer Society, pp. 684-693, 2003.