Modeling Web Application Architectures with UML

Jim Conallen

Rational Software White Paper

TP 157, 6/99

Uma versão deste material aparece na edição
de outubro de 1999 (volume 42, número 10)
de Communications of the ACM.

Rational®
the software development company

Rational®

Índice Analítico

Resumo

Os aplicativos da Web estão se tornando cada vez mais complexos e essenciais para a missão. Para ajudar a gerenciar essa complexidade, eles precisam ser modelados. A UML (Unified Modeling Language) é a linguagem padrão para a modelagem dos sistemas intensivos do software. Ao tentar modelar os aplicativos da Web com UML, fica aparente que alguns componentes não se encaixam perfeitamente nos elementos de modelagem UML padrão. Para trabalhar com uma notação de modelagem para o sistema inteiro (componentes da Web e componentes de nível intermediário tradicional), a UML deve ser estendida. Este artigo apresenta uma extensão da UML, utilizando seu mecanismo formal de extensão. A extensão foi projetada para que componentes específicos da Web possam ser integrados com o restante do modelo do sistema e para exibir o nível de abstração apropriado e os detalhes adequados para os designers, implementadores e arquitetos de aplicativos da Web.

Visão Geral

Um novo termo foi digitado no vocabulário de TI nos últimos anos: Aplicativo da Web. Parece que todos os envolvidos com os sistemas de software de negócios têm planos para a construção de aplicativos da Web com muitas iniciativas de software não relacionadas aos negócios, mas que também estão interessadas. Muitos que logo adotaram o termo aplicativo da Web dessa arquitetura, como os próprios sistemas, desenvolveram de pequenos add-ons de Web sites bem-sucedidos para aplicativos n-tiered robustos. É comum que um aplicativo da Web sirva dezenas de milhares de usuários simultâneos, distribuídos no mundo todo. Arquitetar aplicativos da Web é um negócio sério.

Ao ser testado, o termo aplicativo da Web tem significados levemente diferentes para pessoas diferentes. Alguns acreditam que um aplicativo da Web é nada mais que utilize Java; outros consideram os aplicativos da Web nada mais que um servidor da Web. O consenso geral está em algum lugar entre os dois. Para as nossas finalidades neste documento, definiremos imprecisamente um aplicativo da Web a ser um sistema da Web (servidor da Web, rede, HTTP, navegador) em que a entrada do usuário (navegação e entrada de dados) afeta o estado do negócio. Essa definição tenta estabelecer que um aplicativo da Web é um sistema de software com o estado do negócio e sua "extremidade frontal" é em grande parte fornecida por um sistema da Web.

A arquitetura geral de um aplicativo da Web é a de um sistema de servidor do cliente com algumas notáveis distinções. Uma das vantagens mais significantes de um aplicativo da Web está na sua implementação. Implementar um aplicativo da Web é normalmente uma questão de configurar os componentes laterais do servidor em um rede. Não é necessário um software ou uma configuração especial da parte do cliente. Outra diferença significante é a natureza da comunicação entre o cliente e o servidor. O protocolo de comunicação principal de um aplicativo da Web é HTTP, que é um protocolo sem conexão que foi projetado para ter vigor e tolerância a falhas em vez de rendimento máximo do processamento de comunicação. A comunicação entre um cliente e um servidor em um aplicativo da Web normalmente gira em torno da navegação de páginas da Web, não as comunicações diretas entre o lado servidor e os objetos do lado cliente. Em um nível de abstração, todas as mensagens em um aplicativo da Web podem ser descritos como o pedido e a recepção de entidades de página da Web. Geralmente falando, a arquitetura de um aplicativo da Web não é muito diferente de um Web site dinâmico.

As diferenças entre um aplicativo da Web e um Web site, mesmo que seja um dinâmico, envolve o uso. Os aplicativos da Web implementam a lógica de negócios e sua utilização altera o estado do negócio (conforme capturado pelo sistema). Isso é importante porque define o foco do esforço de modelagem. Os aplicativos da Web executam a lógica do negócio e assim os modelos mais importantes do sistema enfatizam a lógica e o estado do negócio e não os detalhes da apresentação. A apresentação é importante (caso contrário, o sistema não faria bem a ninguém), entretanto, você deve tentar uma separação clara entre o negócio e a apresentação. Se os problemas da apresentação forem importantes ou até mesmo complicados, eles também devem ser modelados, mas não necessariamente como uma parte integral do modelo lógico do negócio. Adicionalmente os recursos que funcionam na apresentação tendem a ser mais artísticos e menos preocupados com a implementação das regras de negócios.

Uma metodologia/notação associada ao desenvolvimento dos Sistemas da Web é a RMM (Relationship Management Methodology). RMM é uma metodologia para o design, construção e manutenção dos sistemas da Web de intranet e Internet. Sua meta principal é reduzir os custos de manutenção de Web sites dinâmicos, dirigidos aos bancos de dados. Defende uma representação visual do sistema para facilitar as discussões sobre design. É um processo iterativo que inclui a decomposição dos elementos visuais nas páginas da Web e sua associação às entidades de banco de dados. RMM é uma abordagem do tipo “sopa de letrinhas” para a criação de manutenção de Web sites dinâmicos.

RMM diminui ao construir aplicativos da Web. Os aplicativos da Web, centrados em lógica de negócios, incluem vários mecanismos tecnológicos para implementação da lógica de negócios que não esteja coberta adequadamente pela notação RMM. Tais tecnologias como o script do lado cliente, applets e controles ActiveX freqüentemente fazem contribuições

significantes para a execução das regras de negócios do sistema. Além disso, os aplicativos da Web podem ser utilizados como um mecanismo de entrega para um sistema de objeto distribuído. Os applets e os controles ActiveX podem conter componentes que interagem assíncronicamente com os componentes do lado servidor via RMI ou DCOM, independente do servidor da Web. Os aplicativos sofisticados também utilizam várias instâncias do navegador e vários quadros no cliente, que estabelecem e mantêm seus próprios mecanismos de comunicação.

Já que todos esses mecanismos contribuem para a lógica de negócio do sistema, eles precisam ser modelados como tal. Além disso, eles precisam ser integrados com o restante dos modelos do sistema porque representam somente parte da lógica de negócio. Em muitas situações, a maior parte da lógica de negócio é executada atrás do servidor da Web em uma das camadas do lado servidor. A escolha da linguagem de modelagem e da notação é normalmente decidida pelas necessidades deste lado do aplicativo. Com a aceitação da UML pelo OMG como uma linguagem de modelagem do objeto oficial, mais e mais sistemas estão sendo expressos com notação UML. Para muitos, a UML é uma linguagem de escolha para modelagem de sistemas de software intensivos. Os problemas principais na modelagem dos aplicativos da Web tornam-se: “Como expressar a lógica de negócios executada em meus componentes específicos da Web junto com o restante do meu aplicativo?” A resposta está em nossa habilidade de expressar a execução da lógica de negócios do sistema nos elementos e nas tecnologias específicos da Web com UML.

Esse documento funciona como uma introdução para os problemas e possíveis soluções para modelagem de aplicativos da Web. Enfatiza os componentes significantes arquitetônicamente específicos para os aplicativos da Web e como modelá-los com UML. Presume-se que o leitor esteja familiarizado com UML, proprietários orientados para os objetos e o desenvolvimento de aplicativo da Web. O trabalho descrito neste documento tem como base algumas suposições inofensivas:

- Os aplicativos da Web são sistemas intensivos de software que estão se tornando mais complexos e estão inserindo a si mesmos em funções críticas da missão.
- Uma maneira de gerenciar a complexidade em sistemas de software é abstraí-los e modelá-los.
- Normalmente, um sistema de software tem vários modelos, cada um representando um ponto de vista, um nível de abstração e um detalhe diferente.
- O nível apropriado de abstração e de detalhe depende dos artefatos e das atividades no processo de desenvolvimento.
- A linguagem de modelagem padrão para os sistemas intensivos de software é a UML (Unified Modeling Language).

Um tratamento mais completo dos conceitos e das idéias expressos neste documento está sendo desenvolvido em um livro de lançamento: “Building Web Applications with UML” que deve ser publicado na Object Technology Series pela Addison Wesley Longman no fim deste ano.

Modelagem

Os modelos ajudam a entender o sistema, simplificando alguns detalhes. A escolha do que deve ser modelado tem um efeito enorme no entendimento do problema e na forma de solução. Os aplicativos da Web, como outros sistemas intensivos de software, normalmente são representados por um conjunto de modelos: modelo de caso de uso, modelo de implementação, modelo de segurança e assim por diante. Um modelo adicional utilizado exclusivamente por sistemas da Web é o mapa do site, uma abstração das rotas de navegação e das páginas da Web em todo o sistema.

A maioria das técnicas de modelagem de hoje estão bem adaptadas ao desenvolvimento de vários modelos de um aplicativo da Web e não precisam de discussão adicional. No entanto, um modelo muito importante como o ADM (Analysis/Design Model) apresenta algumas dificuldades quando uma tentativa de incluir páginas da Web é feita e o código executável associado acompanha os outros elementos no modelo.

Ao decidir como modelar alguma coisa, determinar o nível correto de abstração e de detalhe é essencial para fornecer algo que beneficiará os usuários do modelo. Geralmente falando, é melhor modelar os artefatos do sistema —as entidades da vida real que serão construídas e manipuladas para produzir o produto final. A modelagem da parte interna do servidor da Web ou dos detalhes do navegador da Web não vai ajudar os designers e os arquitetos de um aplicativo da Web. A modelagem das páginas, seus links e todo o conteúdo dinâmico que foi utilizado na criação das páginas e o conteúdo dinâmico das páginas uma vez no cliente é importante—muito importante. São esses artefatos que os designers projetam e os implementadores implementam. As páginas, hyperlinks e o conteúdo dinâmico no cliente e no servidor precisam ser modelados.

A etapa seguinte é o mapeamento desses artefatos para a modelagem de elementos. Os hyperlinks, por exemplo, mapeiam naturalmente para os elementos de associação no modelo. Um hyperlink representa um caminho navegacional de uma página para outra. Estendendo esse pensamento, as páginas podem mapear para classes na visualização lógica do modelo. Se uma página da Web fosse uma classe no modelo, então os scripts da página mapeariam naturalmente para as operações da classe.

Quaisquer variáveis em escopo de página nos scripts mapeariam para atributos de classe. Um problema surge ao considerar que uma página da Web pode conter um conjunto de scripts que são executados no servidor (preparando o conteúdo dinâmico da página) e um conjunto completamente diferente de scripts que são executados somente no cliente (ou seja, JavaScript). Neste cenário, ao olharmos para uma classe de página da Web no modelo, fica confuso se as operações, atributos e até mesmo relacionamentos estão ativos no servidor (enquanto a página está sendo preparada) e quais estão ativos quando o usuário está interagindo com a página no cliente. Além disso, uma página da Web, conforme fornecido em um aplicativo da Web, é realmente melhor modelada como um componente do sistema. Simplesmente mapear uma página da Web para uma classe UML não ajuda a entender melhor o sistema.

Os criadores da UML perceberam que sempre haveria situações em que a UML, fora da caixa, não seria suficiente para capturar a semântica relevante de um domínio ou de uma arquitetura específicos. Para abordar esta finalidade, um mecanismo de extensão formal foi definido para permitir que os profissionais estendam a semântica da UML. O mecanismo permite definir *estereótipos*, *valores marcados* e *restrições* que podem ser aplicados para modelar elementos.

Um *estereótipo* é um adorno que permite definir um novo significado semântico para um elemento de modelagem. *Os valores marcados* são pares de valor chave que pode ser associados a um elemento de modelagem que permite "marcar" qualquer valor como um elemento de modelagem. *As restrições* são regras que definem a boa formação de um modelo. Elas podem ser expressas como um texto de forma livre ou com uma OCL (Object Constraint Language) mais formal.

O trabalho discutido neste documento introduz uma extensão para UML dos aplicativos da Web. Esta extensão, na sua totalidade, está além do escopo deste documento, no entanto, a maioria dos conceitos e explicações é discutida aqui.

Um ponto final na modelagem—de uma distinção muito clara precisa ser colocado entre a lógica de negócios e a lógica de apresentação. Para o aplicativo de negócios típico, somente a lógica de negócios deve fazer parte de ADM. Os detalhes da apresentação como botões animados, ajuda fly over e outros aprimoramentos UI normalmente não pertencem ao ADM. Se um modelo UI separado for construído para o aplicativo, este é o lugar dessas coisas. O ADM precisa permanecer centralizado na expressão do problema do negócio e no espaço da solução. Nesta época de artistas da Web, a aparência e comportamento de uma página da Web é melhor projetada e implementada por um especialista (artista gráfico técnico) e não pelo desenvolvedor tradicional.

Arquitetura de Aplicativo da Web

A arquitetura básica de um aplicativo da Web inclui navegadores, uma rede e um servidor da Web. Os navegadores solicitam "páginas da Web" do servidor. Cada página é uma mistura de instruções de conteúdo e de formatação, expressa em HTML. Algumas páginas incluem scripts do lado cliente que são interpretados pelo navegador. Esses scripts definem o comportamento dinâmico adicional para a página de exibição e freqüentemente interagem com o navegador, o conteúdo da página e os controles adicionais (Applets, controles ActiveX e plug-ins) contidos na página. O usuário visualiza e interage com o conteúdo da página. Às vezes, o usuário digita informações nos elementos de campo na página e as envia para o servidor para processamento. O usuário também pode interagir com o sistema, navegando para páginas diferentes via hyperlinks. Em ambos os casos, o usuário está fornecendo a entrada para o sistema que pode alterar o "estado de negócio" do sistema.

A partir da perspectiva do cliente, a página da Web está sempre em documento HTML formatado. Entretanto, no servidor, uma "página da Web" pode se manifestar de várias formas diferentes. Nos primeiros aplicativos da Web, as páginas dinâmicas da Web foram construídas com a CGI (Common Gateway Interface). A CGI define uma interface para scripts e módulos compilados a serem utilizados para obter acesso às informações passadas junto com um pedido de página. Em um sistema com base em CGI, um diretório especial é normalmente configurado no servidor da Web para conseguir executar scripts em resposta aos pedidos de página. Quando um script CGI é pedido, o servidor, em vez de apenas retornar o conteúdo do arquivo (como seria com qualquer arquivo HTML formatado), processa ou executa o arquivo com o interpretador apropriado (normalmente um shell PERL) e envia a saída de volta para o cliente solicitante. O resultado mais atual desse processamento é um fluxo HTML formatado que é enviado de volta para o cliente solicitante. A lógica de negócios é executada no sistema durante o processamento do arquivo. Durante esse período, ela tem o potencial de interagir com os recursos do lado servidor como banco de dados e os componentes da camada do meio.

Os servidores da Web de hoje foram aprimorados com base neste design básico. Hoje eles têm muito mais consciência da segurança e incluem recursos como o gerenciamento do estado do cliente no servidor, a integração do processamento da transação, a administração remota e o pooling de recursos. Coletivamente a geração mais recente de servidores da Web está abordando os problemas importantes para os arquitetos de missões críticas, escaláveis e aplicativos robustos.

Ao verificar a função dos scripts CGI, os servidores da Web de hoje podem ser divididos em três principais categorias: páginas em script, páginas compiladas e um híbrido das duas. Na primeira categoria, cada página da Web que um navegador do cliente pode pedir está representada no sistema de arquivos do servidor da Web como um arquivo em script. Esse arquivo

é normalmente uma mistura de HTML e alguma outra linguagem de script. Quando a página é solicitada, o servidor da Web delega o processamento dessa página para um mecanismo que o reconheça, com o resultado mais atual que um fluxo HTML formatado recebe para o cliente solicitante. Exemplos disso são: Microsoft's Active Server Pages, Java Server Pages e Cold Fusion.

Na segunda categoria, as páginas compiladas, o servidor da Web carrega e executa um componente binário. Esse componente, como as páginas em script, tem acesso a todas as informações que acompanham o pedido de página (valores de campos de formulário e parâmetros). O código compilado utiliza os detalhes de pedido e normalmente acessa os recursos do lado servidor para produzir o fluxo HTML retornado para o cliente. Embora não seja uma regra, as páginas compiladas tendem a incluir uma funcionalidade maior do que as páginas em script. Uma funcionalidade diferente pode ser obtida ao transmitir parâmetros para o pedido de página compilada. Na verdade, qualquer componente compilado pode incluir toda a funcionalidade das páginas em script de um diretório inteiro. As tecnologias que representam este tipo de arquitetura são: ISAPI da Microsoft e NSAPI do Netscape.

A terceira categoria representa páginas em script que, uma vez solicitadas, são compiladas e essa versão compilada é utilizada posteriormente por todos os pedidos subseqüentes. Somente quando o conteúdo da página original é alterado, enquanto a página é submetida a outra compilação. Esta categoria é um compromisso entre a flexibilidade das páginas em script e a eficiência das páginas compiladas.

Modelagem de Páginas da Web

As páginas da Web, em script ou compiladas, mapeie um a um os componentes na UML. Um componente é uma parte "física" e substituível do sistema. A Visualização de Implementação (Visualização de Componente) do modelo descreve os componentes do sistema e seus relacionamentos. Em um aplicativo da Web, essa visualização descreve todas as páginas da Web do sistema e seus relacionamentos entre si (ou seja, hyperlinks). Em um nível, um diagrama do componente de um sistema da Web é como um mapa do site.

Já que os componentes representam somente o empacotamento físico das interfaces, eles não são adequados para a modelagem das colaborações dentro das páginas. Esse nível de abstração, extremamente importante para o designer e o implementador, ainda precisa ser parte do modelo. Inicialmente, poderíamos dizer que cada página da Web é uma classe UML na Visualização de Design (Visualização Lógica) do modelo e que seus relacionamentos com outras páginas (associações) representam hyperlinks. No entanto, essa abstração é quebrada ao considerar que a página da Web pode potencialmente representar um conjunto de funções e colaborações que existem somente no servidor e que um conjunto completamente diferente pode existir somente no cliente. Qualquer página da Web em script do servidor que empregue HTML Dinâmico (script do lado cliente) como parte da saída é um exemplo dessa página. A reação automática para este problema pode ser estereotipar cada atributo ou operação na classe para indicar se era ou não válido no servidor ou no lado cliente. Neste ponto, nosso modelo, originalmente com a pretensão de ajudar a simplificar as coisas, vai ficando bastante complexo.

Uma melhor abordagem do problema é considerar o princípio da "separação de interesses". Logicamente falando, o comportamento de uma página da Web no servidor é completamente diferente do comportamento no cliente. Ao executar no servidor, é possível ter acesso aos (ou seja, relacionamentos com) recursos do lado servidor (componentes da camada do meio, bancos de dados, sistema de arquivo e assim por diante). Essa mesma página ou a saída HTML de fluxo dessa página no cliente tem um comportamento e um conjunto de relacionamentos completamente diferentes. No cliente, uma página de script tem relacionamentos com o navegador em si via DOM (Document Object Model) e com Java Applets, controles ActiveX ou plug-ins que a página especifica. Para o designer sério, pode haver relacionamentos adicionais com outras páginas "ativas" no cliente que aparecem em outro quadro HTML ou instância do navegador.

Separando os interesses, podemos modelar o aspecto do lado servidor de uma página da Web com uma classe e o aspecto do lado cliente com outra. Distinguimos as duas utilizando o mecanismo de extensão de UML para definir estereótipos e ícones para cada—«página do servidor» e «página do cliente». Os estereótipos em UML permitem definir novas semânticas para um elemento de modelagem. As classes estereotipadas podem ser feitas em um diagrama UML com um ícone personalizado ou simplesmente adornadas com o nome do estereótipo entre sinais de maior e menor («»). Os ícones são úteis para os diagramas de visão geral em que a utilização de marcações simples é melhor quando os atributos de classe e as operações são expostos.

Para as páginas da Web, os estereótipos indicam que a classe é uma abstração do comportamento lógico de uma página da Web no cliente ou no servidor. As duas abstrações estão relacionadas uma com a outra com um relacionamento direcional entre as duas. Essa associação é estereotipada como «construção», já que pode-se dizer que uma página do servidor constrói uma página do cliente (Figura 1). Todas as páginas dinâmicas da Web (ou seja, páginas cujo conteúdo é determinado no tempo de execução) são construídas com uma página do servidor. Todas as páginas do cliente são construídas, na maioria das vezes, por uma única página do servidor; entretanto, é possível que uma página do servidor construa várias páginas do cliente.

Um relacionamento comum entre as páginas da Web é o hyperlink. Um hyperlink em um aplicativo da Web representa um caminho de navegação pelo sistema. Esse relacionamento é expresso no modelo com uma associação estereotipada do «link». Essa associação sempre tem origem em uma página do cliente e aponta para um cliente ou uma página do servidor.

Os hyperlinks são implementados no sistema como um pedido para uma página da Web e as páginas da Web são modeladas como componentes na Visualização de Implementação. Uma associação do link para uma página do cliente é, na maior parte, equivalente a uma associação do link para a página do servidor que constrói a página do cliente. Isso ocorre porque, na verdade, um link é um pedido para uma página, não das abstrações de classe. Já que um componente de página da Web realiza ambas as abstrações de página, um link para qualquer uma das classes realizado pelo componente de página é equivalente.

Os valores marcados são utilizados para definir os parâmetros que são transmitidos junto com um pedido de link. O valor marcado da associação do «link» “Parâmetros” é uma lista de nomes de parâmetros (e valores opcionais) que são esperados e utilizados pela página do servidor que processa o pedido. Na Figura 2, a página ResultadosdeProcura contém um número variável de hyperlinks (0..*) para a página do servidor ObterProduto em que cada link tem um valor diferente para o parâmetro do Iddoproduto. A página ObterProduto constrói a página DetalhesdoProduto do produto especificado pelo parâmetro Iddoproduto.

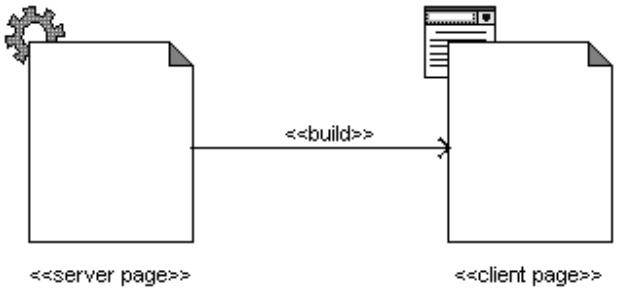

Figura 1. As páginas do servidor constroem as páginas do cliente

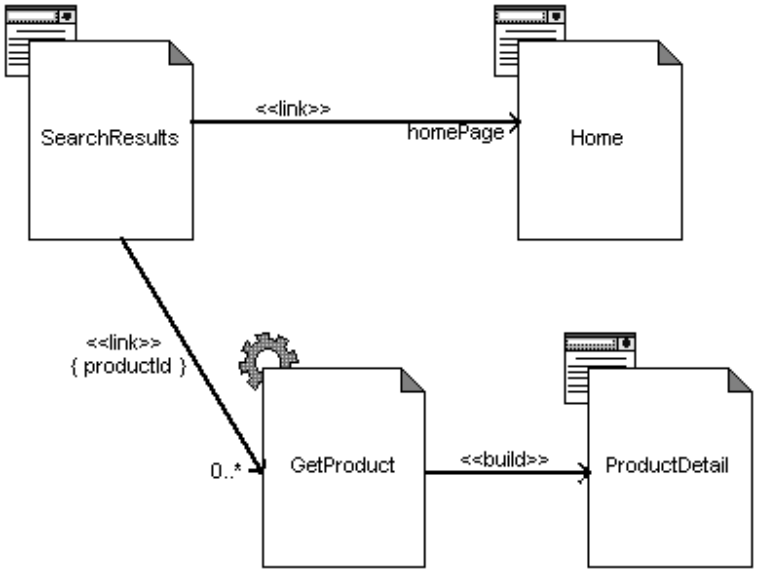

Figura 2. Utilizando parâmetros de hyperlink

Utilizando esses estereótipos, fica mais fácil modelar os scripts e os relacionamentos de uma página. As operações de classe da «página do servidor» tornam-se funções nos scripts do lado servidor da página e seus atributos se tornam variáveis de escopo da página (globalmente acessíveis pelas funções de página). As operações de classe e os atributos da «página do cliente» tornam-se funções e variáveis visíveis no cliente. A vantagem principal de separar os aspectos do lado servidor e cliente de uma página em classes diferentes está nos relacionamentos entre as páginas e outras classes do sistema. As páginas do cliente são modeladas com relacionamentos para os recursos do lado cliente: DOM, Java Applets, controles ActiveX e plug-ins (Figura 3). As páginas do servidor são modeladas com relacionamentos para recursos do lado servidor—componentes da camada do meio, componentes de acesso ao banco de dados, sistema operacional do servidor e assim por diante, ilustrados na Figura 4.

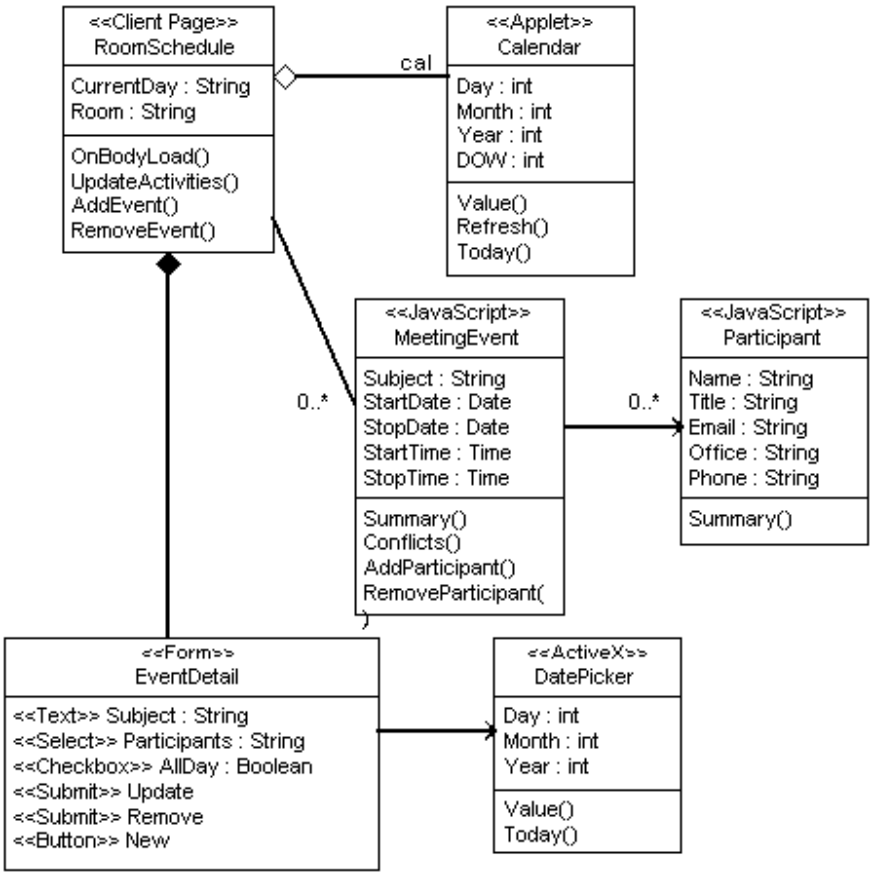

Figura 3. Colaborações do Cliente

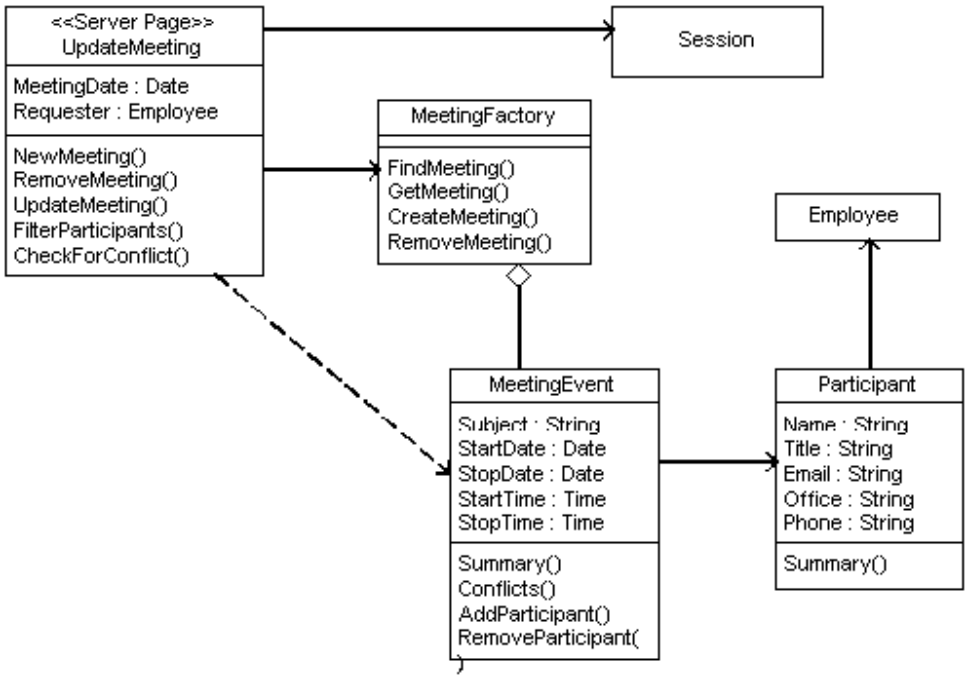

Figura 4. Colaborações do Servidor

Uma das maiores vantagens de utilizar os estereótipos da classe para modelar os comportamentos lógicos das páginas da Web é que as suas colaborações com os componentes do lado servidor possam ser expressas da mesma maneira que qualquer outra colaboração do lado servidor. A «página do servidor» é simplesmente outra classe que participa da lógica de negócios do sistema. Em um nível mais conceitual, as páginas do servidor normalmente controlam a função dos controladores, orquestrando a atividade do objeto de negócios necessária para completar as metas de negócios iniciadas pelo pedido de página do navegador.

No lado cliente, as colaborações podem ficar um pouco complicadas. Isso ocorre, em parte, por causa da variedade de tecnologias que podem ser empregadas. Uma página do cliente mais simples é um documento HTML que contém as informações de conteúdo e de apresentação. Os navegadores fazem páginas HTML utilizando as instruções de formatação na página, às vezes, com folhas de estilo separadas. No modelo lógico, esse relacionamento pode ser expresso com uma dependência de uma página do cliente para uma classe estereotipada da «Folha de Estilo». No entanto, as folhas de estilo são principalmente uma questão de apresentação e são freqüentemente deixadas de fora do ADM

Formulários

O mecanismo principal de entrada de dados para as páginas da Web é o Formulário. Os formulários são definidos em um documento HTML com marcações <form>. Cada formulário especifica a página para a qual deve ser enviado. Um formulário contém um número de elementos de entrada, todos expressos como marcações HTML. As marcações mais comuns são <input>, <select> e <textarea>. A marcação de entrada é de algum modo sobrecarregada já que pode ser um campo de texto, uma caixa de opções, um botão de opções, um botão de comando, uma imagem, um campo oculto assim como outros tipos menos comuns. Os formulários de modelagem significam outro estereótipo de classe: «Formulário». Um «Formulário» não tem operações, já que qualquer operação que possa ser definida em uma marcação <form> é realmente de propriedade da página do cliente. Os elementos de entrada do formulário são atributos estereotipados da classe «Formulário». Um «Formulário» pode ter relacionamentos com Applets ou controles ActiveX que agem como controles de entrada. Cada formulário também tem um relacionamento com uma página do servidor—a página que processa o envio do formulário. Esse relacionamento é estereotipado «enviar». Desde que os formulários estejam completamente contidos em um documento HTML, eles são expressos em um diagrama UML com uma forte forma de agregação. A Figura 5 mostra uma página de carrinho de compras simples que define um formulário e mostra o relacionamento de envio com a página do servidor de processamento.

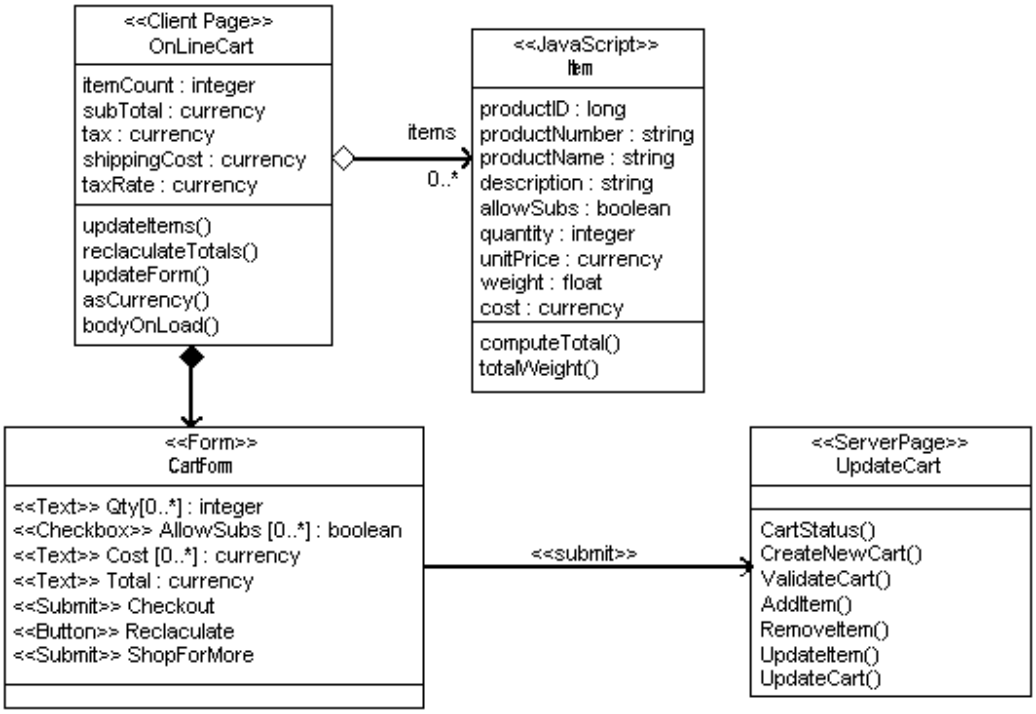

Figura 5. Envio de formulários para as páginas do servidor

Na Figura 5, a classe estereotipada «JavaScript» é um objeto que representa os itens no carrinho de compras. A sintaxe de matriz é utilizada na descrição das propriedades do formulário para os campos que possuem um número variável de instâncias. No caso desse carrinho de compras, significa que o carrinho pode ter de zero a muitos itens, cada um com um elemento de Quantidade, PermitirAss, Custo e Total <entrada>.

Já que toda a atividade na página do cliente é executada com JavaScript e JavaScript é uma linguagem sem tipo, os tipos de dados especificados para qualquer um desses atributos são utilizados somente para esclarecimento do implementador. Quando implementado em JavaScript ou como marcações de entrada HTML, o tipo é ignorado. Isso também se aplica aos parâmetros de função, que embora não sejam completamente exibidos nesta figura, fazem parte do modelo.

Quadros

A utilização de Quadros HTML em um Web site ou aplicativo tem sido assunto de debates polarizados desde a introdução. Os quadros permitem que várias páginas estejam ativas e visíveis para o usuário a qualquer momento. O conjunto de recursos mais recente para os navegadores mais comuns hoje também permite que várias instâncias do navegador estejam ativas na máquina do usuário. Utilizando scripts e componentes HTML Dinâmico, essas páginas podem interagir umas com as outras. O potencial para interações complexas no cliente é significante e a necessidade para modelar ainda melhor.

O arquiteto de software decide se os quadros ou várias instâncias do navegador serão empregados em um aplicativo. Se for assim, pelos mesmos motivos de anteriormente, o modelo de comportamento desse lado cliente precisa ser representado no ADM. Para modelar o uso do quadro, definimos mais dois estereótipos de classe—«frameset » e «destino»—e um estereótipo de associação do «link de destino ». Uma classe de frameset representa um objeto Contêiner e mapeia diretamente para a marcação HTML <frameset>. Contém páginas do cliente e destinos. Uma classe de destino é um quadro nomeado ou uma instância do navegador que é utilizada como referência por outras páginas do cliente. Uma associação de link de destino é um hyperlink para outra página, mas é realizado em um destino específico. No exemplo mostrado na Figura 6, uma visualização de contorno comum é apresentada em um navegador utilizando dois quadros. Um quadro é nomeado com um destino (Conteúdo) em que o outro quadro simplesmente contém uma página do cliente. Este quadro da página do cliente é o TOC (índice) do manual. Os hyperlinks nesta página são destinados para que sejam executados no quadro Conteúdo. O efeito é um índice estático no lado esquerdo e o conteúdo do manual, capítulo por capítulo, na página à direita.

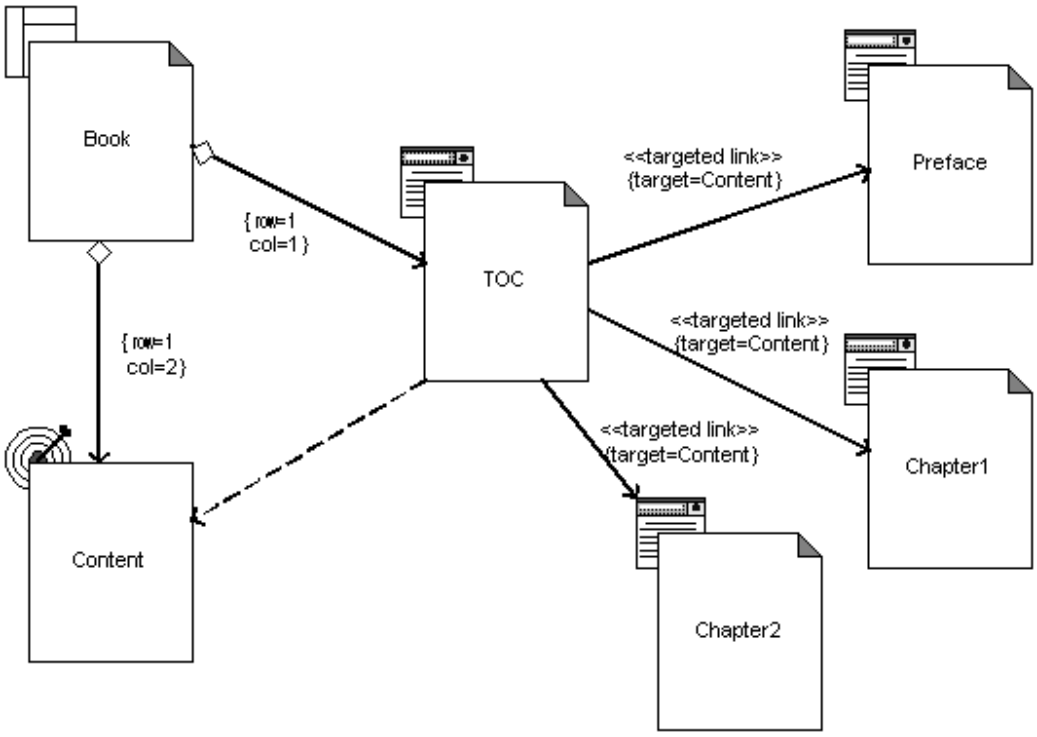

Figura 6. Exemplo de Quadros

Muitas das especificações da apresentação atual são capturadas por valores marcados no frameset e nas associações. Dois valores marcados no relacionamento de agregação entre um frameset e um destino ou na página do cliente especificam a linha do frameset e a coluna ao qual o destino ou a página pertencem. O valor marcado como "Destino" na associação do link de destino identifica o «destino» em que a página deve ser executada.

Quando um destino não é agregado a um frameset, significa que uma instância separada do navegador é utilizada para executar as páginas. É importante manter em mente que essa notação está expressando uma única instância de uma máquina do cliente. Presume-se que vários destinos independentes estarão em execução na mesma máquina e o diagrama expressa o comportamento do lado cliente da instância de um cliente. Qualquer outra configuração de implementação precisaria ser muito documentada no modelo para melhor compreensão.

Conclusão

As idéias e os conceitos discutidos neste documento são uma introdução para os problemas e as soluções para modelagem dos elementos específicos para aplicativos da Web com UML. O objetivo desse trabalho é apresentar uma forma coerente e completa para integrar a modelagem dos elementos específicos da Web com o restante do aplicativo, de modo que os níveis de detalhe e de abstração sejam apropriados para designers, implementadores e arquitetos de aplicativos da Web. Uma primeira versão de uma extensão formal para a UML dos aplicativos da Web está perto da conclusão. Essa extensão fornecerá uma maneira comum para arquitetos e designers para expressar todo o design dos aplicativos da Web com UML.

As informações mais recentes nessa extensão podem ser encontradas na Internet nas seções Rational Rose e UML de Rational Software's Web site.

Rational®
the software development company

Duas Sedes:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Sem custo: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
Localização Internacional: www.rational.com/worldwide