Rational Unified Process for Systems Engineering

Parte II: Arquitetura do Sistema

por Murray Cantor
Principal Engineer
Rational Brand Services
IBM Software Group

No mês passado começamos uma série de três partes para fornecer uma visão geral da evolução mais recente do Rational Unified Process for Systems Engineering ® ou RUP SE ®. O RUP SE é um aplicativo da estrutura do processo de engenharia de software do Rational Unified Process ® ou RUP ®. Os usuários do RUP devem observar se o RUP Plug-In for SE disponível no momento é o RUP SE v1 Plug-In que foi disponibilizado em 2002.

A Parte I incluiu uma discussão de sistemas, os desafios que o desenvolvedor de sistemas modelos enfrentam e como o RUP SE os determina, as técnicas de especificação de requisitos e modelagem baseadas no RUP SE Unified Modeling Language (UML) e o uso da semântica de UML. Este mês, na Parte II, focalizaremos a arquitetura do sistema e apresentaremos a estrutura da arquitetura do RUP SE, que descreve as partes internas do sistema com base em vários pontos de vista. A Parte III, a ser publicada em outubro, abordará a análise e o fluxo de requisitos, uma introdução ao método para derivar requisitos e especificações para os elementos da estrutura do RUP SE. Isso incluirá uma descrição do Joint Realization Method, uma técnica moderna para derivar juntamente a especificação de elementos de arquitetura em vários pontos de vista. A Parte III também incluirá uma discussão da programática do RUP SE.

Nota do editor: O RUP SE v1 Plug-In se tornou disponível em 2002 e a v2 desse plug-in tornou-se disponível em junho de 2003. Embora as informações nesta série seja consistente com v2, os artigos realmente ocorrem com uma possível extensão à estrutura do processo. Observe que o RUP SE Plug-In, v1 e v2, é transferível por download do IBM Rational Developer Network (autorização requerida).

Definições

Uma clara compreensão do RUP SE é impossível sem um conhecimento básico de vários termos e conceitos. [Outros padrões podem definir esses termos de maneira diferente; o que esperamos disso é consistência interna.]

Decomposição do sistema: A engenharia bem-sucedida do sistema depende da capacidade para considerar muitas coisas ao mesmo tempo. A decomposição do nível do sistema é uma técnica poderosa para realizar isso.

Um sistema pode ser decomposto de duas maneiras:

- Em sistemas adicionais que utilizam decomposição lógica; esta é a chamada decomposição "sistemas de sistemas".
- Em componentes do sistema que compõem o sistema entregue.

Dimensões do modelo do sistema: Um modelo de sistema RUP SE tem duas dimensões, que permitem a separação de interesses por diferentes equipes envolvidas no design e na construção do sistema.

- Dimensão de ponto de vista: o contexto para determinar um conjunto limitado de problemas de qualidade.
- Dimensão a nível de modelo: Diagramas de UML que capturam um nível específico de detalhe de design.

Modelo: Uma representação de um sistema, incluindo visualizações que capturam todas as áreas de interesse, níveis de especificidade e relacionamentos da entidade de modelo.

Nível de modelo: O nível de abstração em que cada modelo pode ser construído, do mais geral, ocultando ou encapsulando detalhes, ao mais específico, expondo decisões de design mais explícitas e detalhadas.

Ponto de Vista: Um ponto de vista, como o nome diz, é uma "posição" nocional a partir da qual alguns aspectos ou preocupações com o sistema ficam visíveis, envolvendo a aplicação de um conjunto de conceitos e regras para formar um filtro conceitual. Para compreender um sistema, geralmente não é suficiente examinar o sistema real em si, que é porque os modelos são construídos para representar os vários pontos de vista envolvidos.

Visualização: Uma projeção de um nível de modelo que mostra as entidades relevantes de um ponto de vista específico. Essas projeções são geralmente ilustradas por diagramas de algum tipo. A interseção do nível (de abstração) do ponto de vista e do modelo conterá (ou pelo menos identifica) visualizações de um ou mais modelos relevantes a esse ponto de vista (preocupação) nesse nível de abstração.

Pontos de Vista

A estrutura do RUP SE fornece um conjunto de pontos de vista, conforme expressado na Tabela 1.

Tabela 1: Pontos de Vista do Modelo de Sistema

Expressões de Ponto de Vista		Preocupação
Trabalhador	Funções e responsabilidades dos trabalhadores do sistema	Atividades do trabalhador Decisões de automação Interação humana/do sistema Especificações de desempenho humano
Lógico	Decomposição lógica do sistema como um conjunto coerente e subsistemas UML que colaboram para proporcionar o comportamento desejado	Funcionalidade adequada do sistema para realizar casos de uso Capacidade de extensão e possibilidade de manutenção Reutilização interna Boa coesão e conectividade
Físico	Decomposição física do sistema e especificação dos componentes físicos	Características físicas adequadas para funcionalidade do host e para atender a requisitos suplementares
Informações	Informações armazenadas e processadas pelo sistema	Capacidade suficiente para armazenar dados Rendimento suficiente Para fornecer acesso Oportuno aos dados

Processo	Encadeamentos de controle, os quais transportam os elementos de computação	Particionamento suficiente de processamento para suportar necessidades de simultaneidade e confiabilidade

Os pontos de vista da Tabela 1 são alguns dos pontos de vista mais comuns para sistemas de software intenso. Muitas arquiteturas de sistema exigem pontos de vista adicionais que são específicos do domínio: proteção, segurança e pontos de vista mecânicos, por exemplo.

Os pontos de vista representam as diferentes áreas de preocupação que devem ser tratadas na arquitetura e no design do sistema. Se houver envolvidos ou especialistas do sistema cujas preocupações sejam importantes para a arquitetura global, provavelmente haverá a necessidade de um conjunto de visualizações para capturar suas decisões de design.

É importante construir uma equipe de arquitetura de sistema com membros cujas habilidades permitirão que eles gerenciem os diversos pontos de vista. A equipe pode ser composta de analistas de negócios e usuários que assumem a responsabilidade primária do ponto de vista do trabalhador, arquitetos de software que atendam ao ponto de vista lógico e engenheiros que se preocupem com o ponto de vista físico, bem como especialistas em pontos de vista específicos do domínio.

Níveis de Modelo

Além dos pontos de vista, a construção de uma arquitetura de sistema requer níveis de especificação. À medida que é desenvolvida, a arquitetura evolui de uma especificação geral e abstrata para uma especificação detalhada e mais específica. Consistentes com as diretrizes do RUP, há quatro níveis de modelo de arquitetura no RUP SE, conforme descrito na Tabela 2.

Tabela 2: Níveis de Modelo do RUP SE

Nível de Modelo	Expressa
Contexto	O sistema e seus agentes
Análise	Particionamento inicial do sistema em cada um dos pontos de vista para estabelecer a abordagem conceitual
Design	Realização do nível de análise para hardware, software e pessoas

Implementação	Realização do modelo de design em configurações específicas

Através desses níveis, o design passa do abstrato para o físico. O nível de modelo de contexto captura todas as entidades externas (agentes) que interagem com o sistema. Esses agentes podem ser externos ou internos para a empresa que implanta o sistema. Em qualquer um dos casos, os agentes podem ser seres humanos ou outros sistemas. No nível de análise, as partições não refletem escolhas de hardware, software e pessoas. Em vez disso, elas refletem abordagens de design para separar o que o sistema precisa fazer e como o esforço deve ser distribuído. No nível de design, as decisões são tomadas quanto aos tipos de componentes de hardware e software e quanto às funções do trabalhador que são necessárias. No nível de implementação, escolhas específicas de tecnologia de hardware e software são feitas para implementar o design. Por exemplo, no nível de design, um servidor de dados é especificado. No nível de implementação, a decisão é tomada para utilizar uma plataforma específica que esteja executando um aplicativo de banco de dados específico.

É importante manter a rastreabilidade entre esses níveis. À medida que a empresa ou a missão é alterada, as visualizações de nível de contexto precisam ser corrigidas, juntamente com todas as visualizações de nível inferior afetadas. Conforme a tecnologia subjacente é alterada, o nível de implementação e, possivelmente, o nível de design podem ser afetados. Em resumo, o impacto das mudanças corporativas fluem de maneira negativa, enquanto o impacto das mudanças tecnológicas fluem de maneira positiva.

Visualizações da Arquitetura do Sistema

A próxima etapa é capturar a arquitetura de sistema em um conjunto de visualizações que expressam a arquitetura de vários pontos de vista e níveis de modelo. Cada uma das células na Tabela 3 fornece uma visualização do sistema. Observe que no nível da implementação, um diagrama único captura a realização dos componentes de hardware e software para cada configuração do sistema.

Tabela 3: Estrutura de Modelo do RUP SE

Níveis de Modelo	Pontos de Vista de Modelo				
	Trabalhador	Lógico	Informações	Físico	Processo
Contexto	Visualização da organização UML	Diagrama de contexto do sistema	Visualização de dados corporativos	Localidade da empresa (distribuição de recursos corporativos)	Processos de negócios
Análise	Visualização Generalizada do trabalhador do sistema	Visualização do subsistema	Visualização de dados do sistema	Visualização de localidade do sistema	Visualização de processo do sistema

Design	Visualização do trabalhador do sistema	Visualizações de classe do subsistema Visualizações de compon. de software	Esquema de dados do sistema	Visualização do nó do descritor	Visualização do processo detalhada
Implementação	Instruções e especificações de função do trabalhador	Configurações: diagrama de implementação com componentes do sistema de hardware e software			

Os relacionamentos entre níveis de modelo, pontos de vista e visualizações podem ser vistos na Figura 1 a seguir.

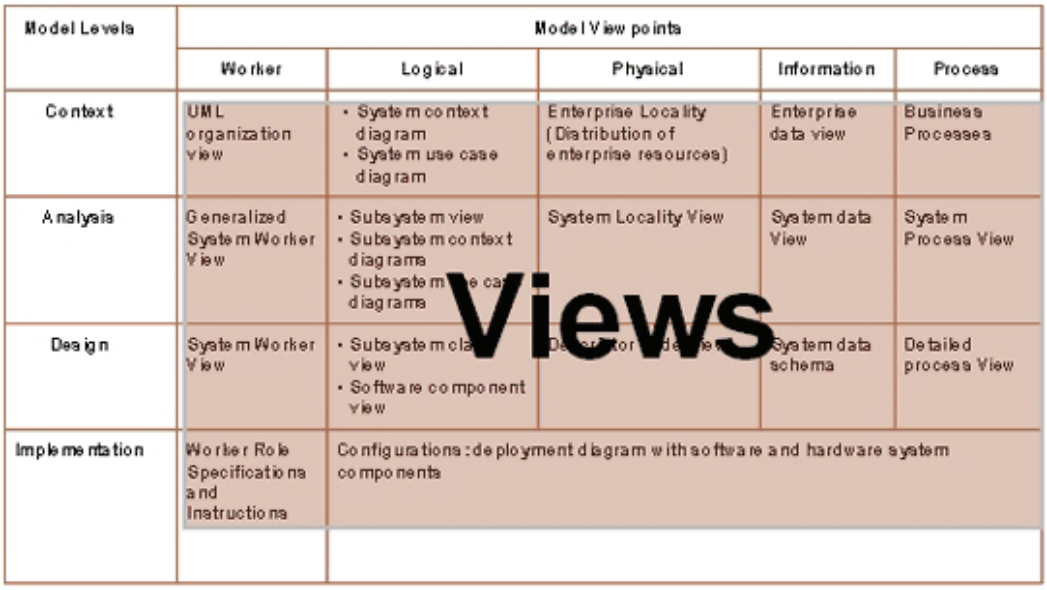

Figura 1: Níveis de Modelo, Pontos de Vista e Visualizações

Os pontos de vista específicos do domínio também precisam ter artefatos no local para um ou mais níveis. O conjunto de artefatos do projeto dentro desta estrutura deve fazer parte do caso de desenvolvimento do projeto. Vamos dar uma rápida olhada em cada um dos pontos de vista.

Nota: Nesta publicação, o UML 2.0 está sendo preparado para adoção e o OMG liberou um Pedido de Proposta para um Perfil de Systems para UML. Depois que o UML 2.0 e o perfil de Systems forem adotados, a semântica de visualização do RUP SE será atualizada para tirar total proveitos de seus padrões.

Ponto de Vista do Trabalhador

Os trabalhadores são suficientemente exclusivos para garantir seus próprios pontos de vista. Trabalhadores são entidades lógicas e físicas. Eles são entidades lógicas uma vez que, quando instruídos, eles podem prestar serviços e colaborar com outras entidades lógicas.

Eles são entidades físicas uma vez que são limitados no que diz respeito a desempenho, pronto atendimento e capacidade. Obviamente, os trabalhadores são entidades de caixa preta e não estão sujeitos à subdivisão adicional no modelo.

A razão sobre como os trabalhadores interagem com as partes automatizadas do sistema e entre si é uma especialidade da engenharia de sistema. O ponto de vista do trabalhador fornece a definição para essa razão.

Além disso, observe que os trabalhadores do sistema não são iguais aos trabalhadores de negócios. Os trabalhadores do sistema são seres humanos que fazem parte do sistema. Eles não são agentes do sistema, uma vez que são parcialmente responsáveis pela entrega dos serviços do sistema. Na estrutura do RUP SE, os trabalhadores do sistema são representados como classes com estereótipo. Eles poderão ser associados se tiverem dependências em um outro relacionamento ou em algum outro relacionamento. Na visualização generalizada do sistema, os trabalhadores genéricos do sistema são expressos com poucos detalhes.

Em alguns aplicativos, é útil apresentar um resumo da parte automatizada do sistema -- a máquina, que difere de um sistema geral em que sua realização não contém trabalhadores. Os trabalhadores generalizados e a máquina podem ser usados no fluxo de trabalho de curso decrescente para determinar as especificações do trabalhador e considerar as decisões de automação. A Figura 2 mostra um exemplo de um diagrama de trabalhador.

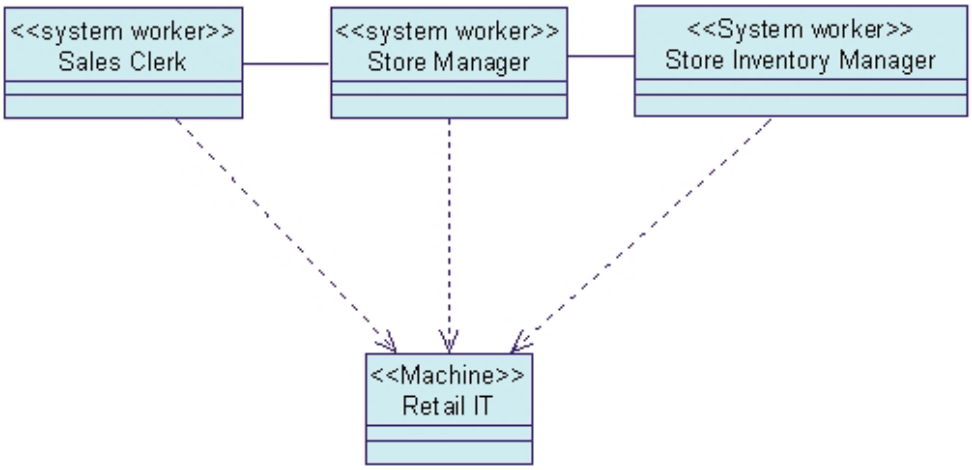

Figura 2: Diagrama de Trabalhador do RUP SE

Por exemplo, se você estiver modelando um sistema para um navio, no nível de modelo da análise você poderá representar um marinheiro como um trabalhador geral do sistema. No nível de design, no entanto, poderá definir uma grande quantidade de funções específicas do marinheiro.

Nota: Você pode desejar incluir um classificador com estereótipo adicional no diagrama do trabalhador -- uma máquina -- para suportar decisões de automação. No método de realização coletiva discutido a seguir, a máquina desempenhará etapas lógicas de caixa branca a serem suportadas pela automação.

Ponto de Vista Lógico

O ponto de vista lógico é o mais familiar para analistas de objetos. Ele descreve, em diferentes níveis de resumo, o tipo de objeto que compreende o sistema. Os elementos das visualizações no ponto de vista lógico são classes e subsistemas de UML. No UML 1.4, os sistemas e subsistemas são herdados de classificadores e pacotes; não há sintaxe de UML que capture os aspectos do classificador e do pacote de um subsistema. Normalmente, no UML, os subsistemas são representados como pacotes com dependências. No RUP e RUP SE, as classes de proxy são utilizadas para representar a semântica do classificador. No RUP SE, criamos estereótipos dos proxies e dos pacotes de quantos sistemas ou subsistemas forem apropriados e, conforme apropriado, adicionamos a semântica do sistema descrita acima a subsistemas. A Figura 3 mostra uma visualização de subsistema UML para um sistema de varejo pela Internet que utiliza a notação comum. O indivíduo poderá escolher utilizar classificadores de subsistema no lugar dos pacotes contidos nesta figura.

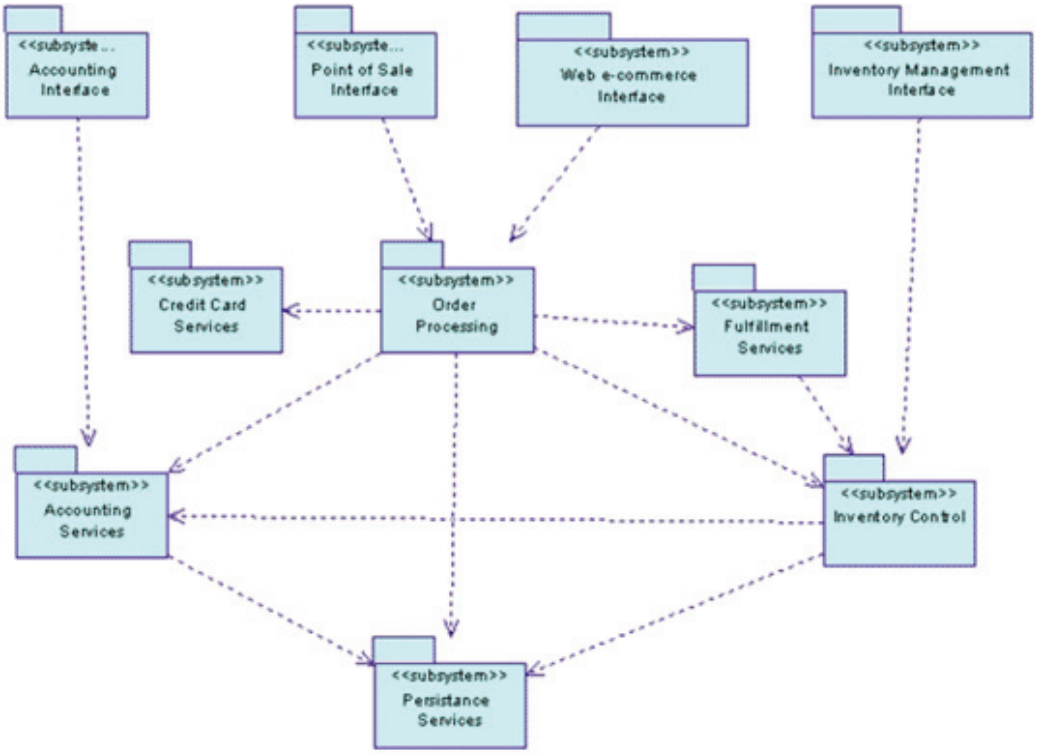

Figura 3: Diagrama de Subsistema de Varejo pela Internet

Clique para expandir

Ponto de Vista Físico

Na engenharia de sistemas, os recursos físicos são uma parte ou um aspecto do sistema. Ela compreende que a semântica precisa ser fornecida para justificar as propriedades dos elementos da realização física do sistema. Mais especificamente, o resultado de um ambiente de engenharia de sistema inclui uma especificação detalhada do hardware a ser criado ou adquirido. Observe que a engenharia de sistema não inclui as disciplinas da engenharia de hardware (mecânica, elétrica), mas inclui a especificação suficiente a ser utilizada como entrada para a equipe de design de hardware.

Como mostrado na Tabela 3, o RUP SE utiliza um nível de análise, o diagrama de ponto de vista físico denominado Visualização de localidade do sistema.

No ponto de vista físico, o sistema é decomposto em elementos que hospedam os serviços do subsistema lógico. Os diagramas de localidade são a expressão mais abstrata desta decomposição. Eles expressam onde o processamento ocorre sem empatar a localidade de processamento a uma localização geográfica específica ou mesmo sem empatar a realização do recurso de processamento a hardware específico. Localidade refere-se à proximidade de recursos, não necessariamente ao local, que é capturado no modelo de design. Por exemplo, uma visualização de localidade poderia mostrar que o sistema permite o processamento de um satélite espacial e uma estação terrestre. O processamento hospedado em cada localidade é uma importante consideração de design.

Os diagramas de localidade mostram o particionamento inicial, como os elementos do físicos do sistema são distribuídos e como eles são conectados. O termo localidade é utilizado porque a localidade de processamento muitas vezes é um problema quando se está considerando principalmente requisitos não funcionais.

Como mostrado na Figura 4, os diagramas de localidade consistem em dois elementos:

- Localidades: agrupamentos de recursos físicos que permitem um particionamento conceitual, físico do sistema. Seu ícone é um cubo arredondado.
- Conexões: vinculações entre as localidades que podem ser utilizadas para transmitir dados, pedidos de serviço ou entidades de E/S. As conexões são representadas em UML como associações com estereótipo.

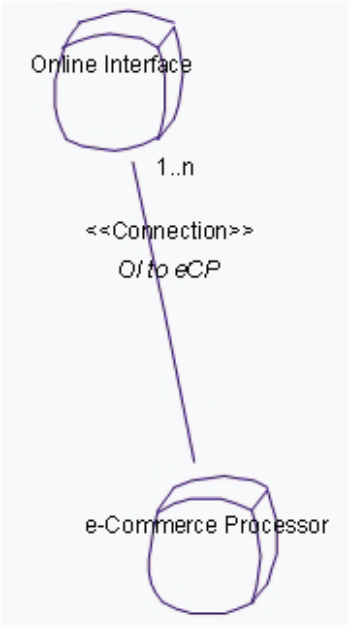

Figura 4: Elementos do Diagrama de Localidade

Semântica de Localidade

As localidades são utilizadas para compreender as características físicas da classe do sistema e suas semânticas derivam daquelas características associadas à natureza física do sistema. Especificamente, as localidades têm atributos de classe e de instância e medidas de eficácia capturadas como valores marcados. As localidades têm dois conjuntos padrão de tags:

- Qualidade: confiabilidade, disponibilidade, desempenho, capacidade e assim por diante
- Gerenciamento: custo e risco técnico

Essas características de localidade formam um conjunto nominal. Cada equipe de desenvolvimento deve determinar o melhor conjunto de características para seus projetos. Essa determinação poderá ser uma atividade de especificação do caso de desenvolvimento.

As características de localidade são configuradas para atender a seus requisitos derivados. Há uma sutil diferença entre características e requisitos. Por exemplo, para obter razões ideais de engenharia, você deve especificar uma localidade que exceda os requisitos.

Na seção sobre Localidades, Serviços e Interfaces a seguir, mostraremos que as localidades hospedam serviços do subsistema.

Semântica de Conexão

As localidades são unidas por conexões, que representam as vinculações físicas entre localidades. Conexões são associações com estereótipo com valores marcados, novamente capturando características. As tags de conexão nominal são:

- Rendimento: taxa de transferência, protocolos suportados
- Gerenciamento: custo, risco técnico

Como as localidades hospedam serviços, as conexões devem transmitir chamadas de serviços. Na realidade, há pelo menos três tipos de fluxo que temos de considerar em sistemas:

- Fluxo de controle
- Fluxo de dados
- Fluxo de materiais

Considere, por exemplo, o acelerador em um automóvel. A ligação do acelerador é a conexão de controle que transmite os pedidos de serviço (abrir ou fechar) ao acelerador. A linha de combustível também é uma conexão com o acelerador. A gasolina em si não é um pedido de serviço, mas uma matéria-prima utilizada pelo acelerador para realizar seus serviços. Finalmente, pode haver uma conexão de dados da rede com o acelerador que contém um fluxo contínuo de dados de status do automóvel e do ambiente que são utilizados para ajustar a resposta ao acelerador.

Localidades e Nós

Lembre-se que os nós de UML são classificadores que possuem recurso de processamento e memória.[1] Utilizada em diagramas de implementação, a semântica do nó UML oferece suporte à compreensão sobre os processadores hosting dos componentes de software. A suposição implícita é que os recursos físicos estão fora do software em consideração. Por exemplo, em engenharia de software, o hardware muitas vezes é visto como uma camada de ativação abaixo do sistema operacional.

O UML fornece artefatos de nível de implementação e design para diagramas de implementação:

- Diagramas de descritor para o nível de design
- Diagramas de instância para o nível de projetar

Especificamente, os diagramas de implementação de instância significam capturar configurações e opções reais de hardware e software, além de fornecer uma base para análise e design do sistema, funcionando como um nível de implementação no ponto de vista físico. O UML Reference Manual descreve uma versão de instância de um diagrama de implementação como "um diagrama que mostra a configuração de nós de processamento de tempo de execução e as instâncias de componentes e objetos que existem neles".2

No RUP SE, esta intenção é preservada. Dessa forma, um nó é um tipo especial de localidade que é utilizado nos níveis de modelo de implementação e de design para especificar recursos físicos que executam software. No entanto, como um tipo de localidade, os nós do RUP SE podem ser esteriotipados para incluir toda a semântica de localidade. Observe que essa semântica difere dos nós padrão no UML. As localidades não são nós muito estereotipados, pois os nós são localidades estereotipadas. O UML 2.0 fornecerá uma melhor maneira de lidar com particionamento físico.

Localidades, Serviços e Interfaces

Uma localidade especifica os recursos físicos que fornecem serviços lógicos. Na prática, cada localidade fornecerá um subconjunto dos serviços de um ou mais dos subsistemas lógicos. A determinação desses serviços é um resultado do fluxo de trabalho de realização coletiva que descreveremos a seguir.

O conjunto de serviços do subsistema hospedado de uma determinada localidade pode ser capturado de algumas maneiras:

- Pesquisa de opinião de documento de serviços do subsistema hospedado
- Interfaces do subsistema associadas

O primeiro método é mais simples, associando um documento de requisitos a uma localidade. O segundo requer um uso mais sofisticado do UML. Os subsistemas são classificadores e seus serviços são operações de classificadores. Além disso, o UML permite que as operações e, portanto, os serviços do subsistema, sejam organizados em interfaces. Dessa forma, uma interface é um subconjunto de serviços do subsistema. Nesta segunda abordagem, um indivíduo define as interfaces necessárias para cada um dos subsistemas e, em seguida, as atribui às localidades apropriadas. Geralmente, haverá mais de uma interface associada a uma localidade.

Negociações de Design

"Negociações de design" é o nome de uma técnica comum da engenharia de sistema: criar um conjunto de abordagens alternativas de design; analisar o custo, a qualidade e a viabilidade das alternativas e, em seguida, escolher a melhor solução.

A visualização de localidade suporta as negociações de design contendo mais de um diagrama de localidade, cada uma representando uma abordagem conceitual diferente para a decomposição física do sistema.

As Figuras 5 e 6 são diagramas de localidade que documentam diferentes abordagens de engenharia para uma empresa de vendas a varejo pela Internet com várias lojas de varejo, armazéns centrais e presença na Web. A primeira solução (Figura 5) descreve a capacidade de processamento nas lojas. A segunda solução (Figura 6) mostra todos os terminais conectados diretamente a um processador do escritório central. Em cada caso, as características podem ser configuradas para as localidades necessárias para realizar o design. Atualmente, a maioria das pessoas concordaria que a Figura 5 representa um melhor design; no entanto, a solução na Figura 6 poderá ser considerada superior em alguns anos.

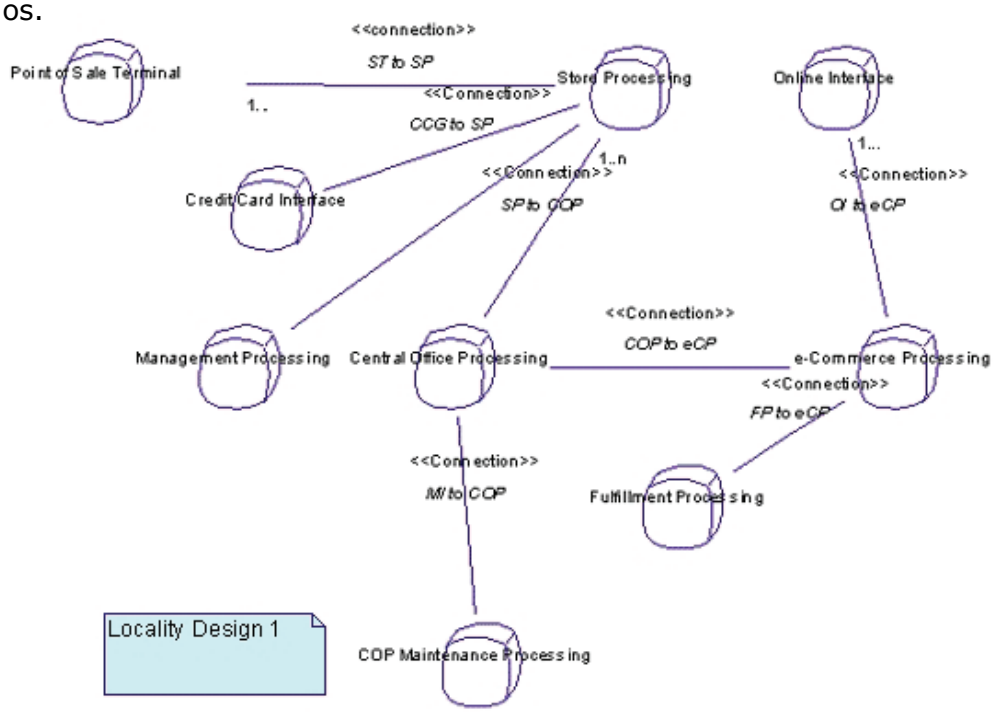

Figura 5: Visualização de Localidade do Sistema -- Exemplo 1

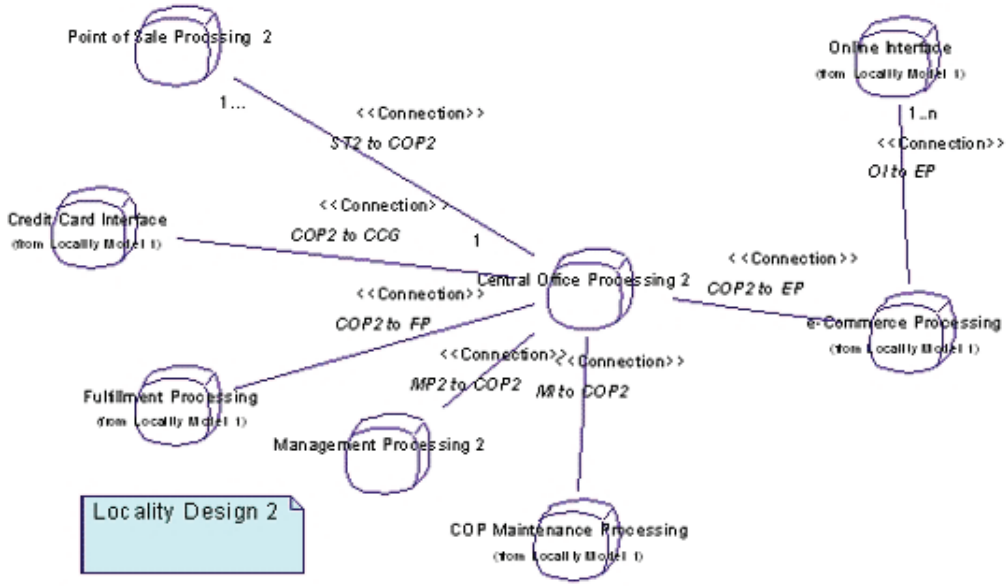

Figura 6: Visualização de Localidade do Sistema -- Exemplo 2

Decomposição e Realização de Localidades

Como os subsistemas, as localidades podem ser decompostas hierarquicamente em localidades adicionais. Está sendo feita uma tentativa de utilizar agregação para associar as localidades às sublocalidades. No entanto, há uma diferença crítica entre o relacionamento total-parcial em uma decomposição física e o relacionamento normalmente expresso com agregação de classe: Em uso comum, quando um objeto de classe (total) é agregado de outros objetos de classe (partes), os atributos do todo incluem os atributos das partes. Os atributos do todo em uma localidade do sistema são funções dos atributos das partes. Um exemplo simples é que o peso do todo é a soma do peso das partes. Muitas vezes, o relacionamento entre um atributo do todo e um atributo da parte é muito mais complexo, mas não há semântica atual no UML para expressar os relacionamentos funcionais entre os atributos. Uma solução alternativa para capturar o relacionamento pode ser inserida no modelo, utilizando atributos e operações privadas que realizam as funções.

Ao realizar localidades como componentes físicos, sugerimos que a realização seja hierárquica. Dessa forma, cada componente faz parte da realização de não mais do que uma localidade. Do contrário, será difícil manter a rastreabilidade de requisitos não funcionais derivados entre as localidades e os componentes. No entanto, você não precisa seguir essa sugestão se desejar ter componentes reutilizáveis em localidades. Neste caso, obviamente, os componentes terão de atender aos requisitos mais severos, descobertos pelo fluxo decrescente dos requisitos do sistema em localidades.

Além disso, em alguns casos, faz sentido criar várias realizações da mesma localidade, fluindo no sentido decrescente para várias implementações do sistema. Por exemplo, um indivíduo pode ter uma linha de produto com diferentes implementações para suportar uma variedade de pontos de preço/desempenho.

Ponto de Vista de Informações

O uso do UML para a modelagem de objeto e de banco de dados relacional é uma prática bem desenvolvida que o RUP SE usa do ponto de vista de informações. Observe que a manutenção da modelagem do banco de dados no modelo do sistema permite a coerência geral do sistema oferecendo suporte a associações entre dados e classes funcionais e atribuindo componentes do banco de dados a localidades.

Ponto de Vista do Processo

O ponto de vista do processo também é representado usando o UML padrão.[3] A Figura 7 mostra um exemplo de uma visualização do processo do sistema.

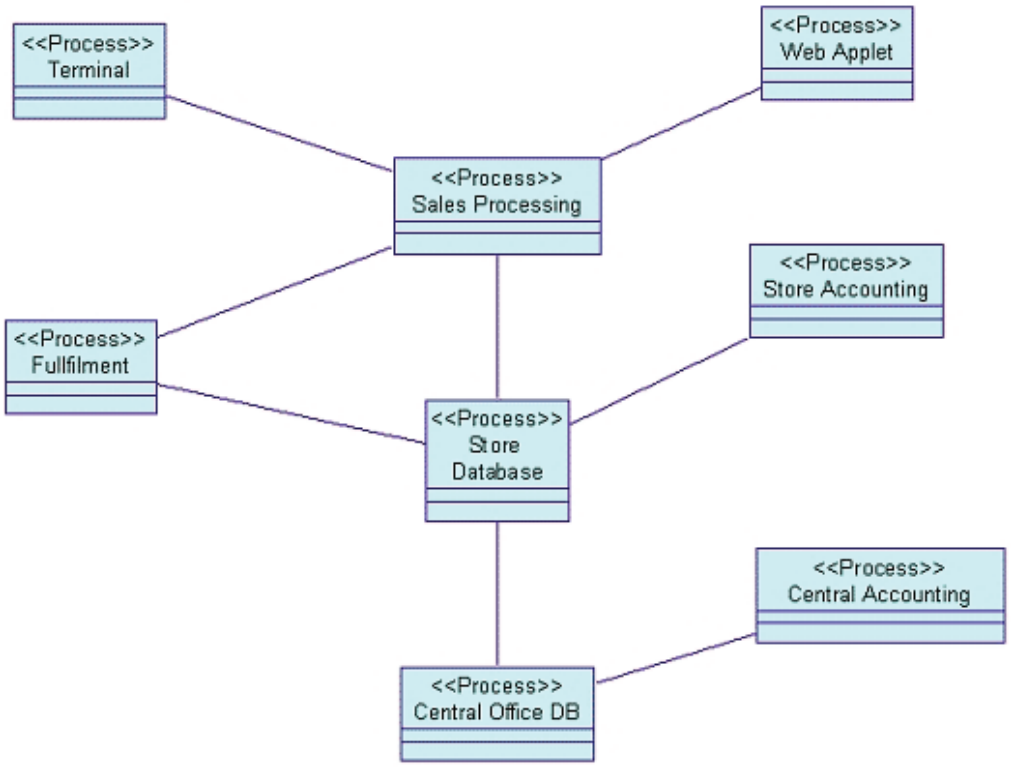

Figura 7: Visualização do Processo do Sistema de Amostra

Movendo entre Níveis de Modelo

A movimentação de níveis de modelo inclui especificidade, não precisão, aos modelos. Em cada nível, você precisa ser o mais preciso possível na especificação de elementos do modelo, pois a precisão em cada nível é incluída na compreensão do sistema e na disciplina do processo. À medida que você se move para baixo nos níveis, cada visualização é uma decisão mais específica, resultando em itens de configuração no nível de implementação. É importante observar que os elementos de modelo em um nível estabeleça os requisitos no próximo nível. Ou então, conforme indicado na Figura 8, podemos dizer que cada nível de modelo realiza requisitos descobertos em um nível superior. Por exemplo,

- O nível de modelo de análise mostra como os requisitos especificados no nível de modelo de contexto são atendidos.
- O nível de modelo de design mostra como os requisitos provenientes do nível de modelo de análise do sistema são atendidos.
- O nível de modelo de implementação atende às especificações de design.

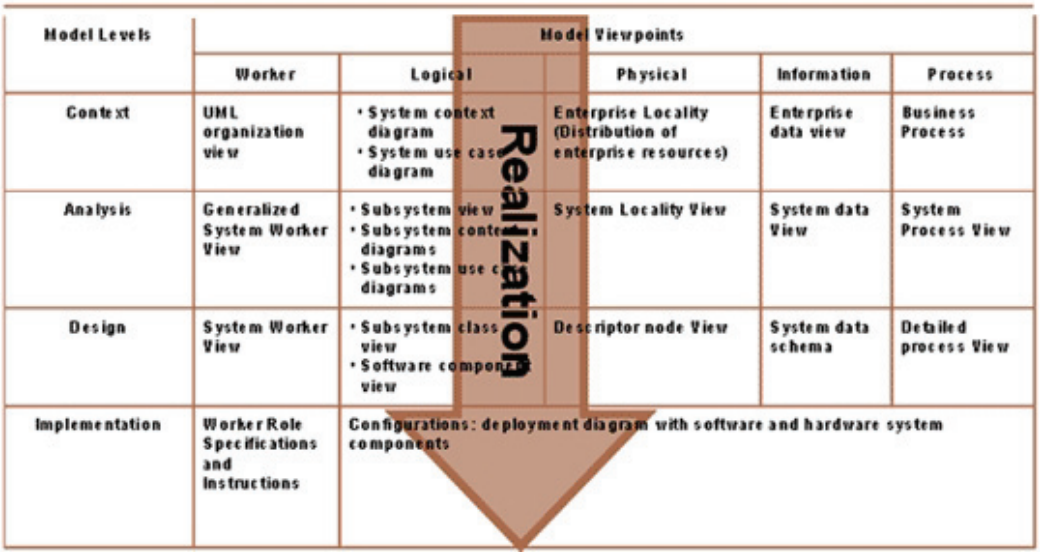

Figura 8: Os níveis de modelo mais baixos realizam requisitos estabelecidos em níveis de modelo mais altos

Clique para expandir

A Figura 9 mostra um exemplo de como o ponto de vista físico no nível de design contém um diagrama de nó de descritor, que mostra um design físico que realiza cada localidade.

Como cada nível de modelo estabelece requisitos ou especificações para serem realizados no próximo nível mais baixo, você pode manter rastreabilidade entre os níveis capturando como os elementos de design atendem a essas especificações.

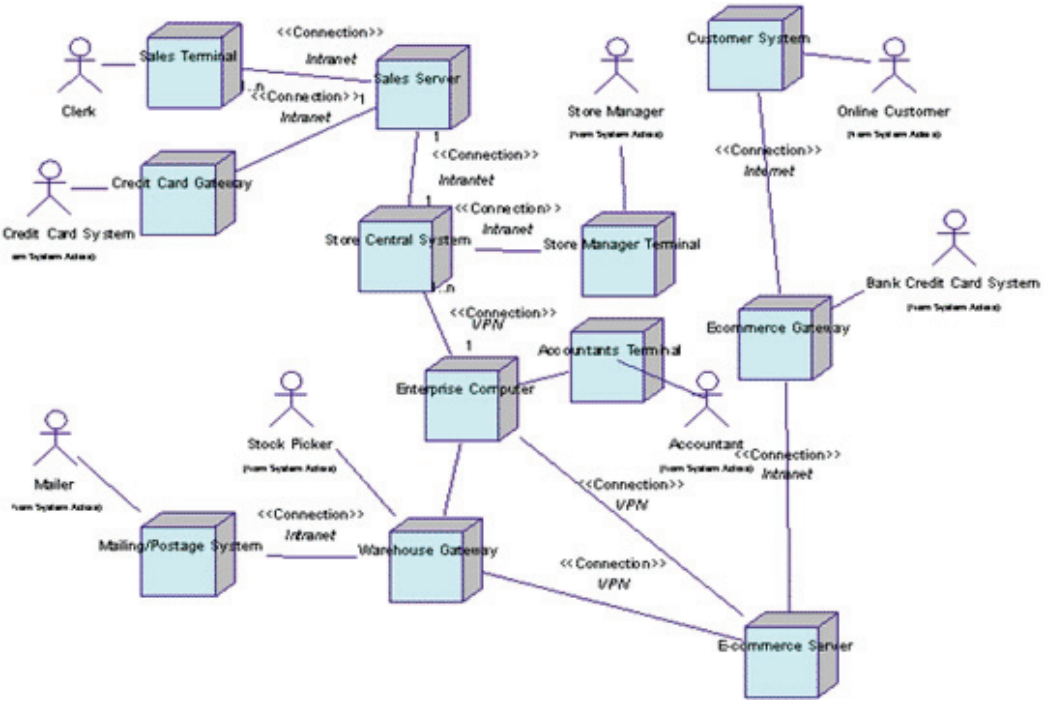

Figura 9: Realização da Visualização da Localidade de Vendas a Varejo pela Internet

Clique para expandir

Na prática, à medida que uma equipe desenvolve um nível de modelo, ela provavelmente poderá descobrir que o nível mais alto deverá ser revisto, pois, por exemplo, um ou mais elementos que ele especificou não pode ser realizado. Portanto, à medida que o desenvolvimento continua, nenhum nível é realmente"congelado"; cada um é mantido em todo o desenvolvimento. No entanto, à medida que o desenvolvimento progride, o foco do esforço geralmente é movido para baixo, nível a nível.

Notas

[1] Grady Booch, James Rumbaugh, and James Jacobson, The Unified Modeling Language User Guide. Addison Wesley, 1999, p.358.

[2] Ibid, p.252ff.

[3] Ibid, p.455.

Para obter mais informações sobre os produtos ou serviços discutidos neste artigo, clique aqui e siga as instruções fornecidas.

Obrigado!