Diagramas de Componentes e Diagramas de Deployment

Ricardo R. Gudwin
05/10/2010

Introdução

Neste texto, apresentamos um resumo da norma UML que descreve diagramas de componentes e diagramas de distribuição (diagramas de deployment, ou as vezes também chamado de diagrama de implantação).

Componentes

A linguagem UML especifica um conjunto de construtos que podem ser utilizados para definir sistemas de software de tamanho e complexidade arbitrários. Em particular, define-se o conceito de **componente**, como uma unidade modular com um conjunto de interfaces bem definidas, que pode ser substituído dentro de seu ambiente. O conceito de componente é oriundo da área de desenvolvimento baseado em componentes, onde um componente é modelado durante o ciclo de desenvolvimento e refinado sucessivamente durante a instalação e execução do sistema.

Um aspecto importante do desenvolvimento baseado em componentes é o reuso de componentes previamente construídos. Um componente pode ser considerado como uma unidade autônoma dentro de um sistema ou subsistema. Ele deve possuir uma ou mais **interfaces**, que podem ser potencialmente disponibilizadas por meio de **portas**, e seu interior é normalmente inacessível. O acesso a um componente deve ocorrer única e exclusivamente por meio de suas interfaces. Apesar disso, um componente pode ser dependente de outros componentes, e a linguagem UML provê mecanismos para representar essa dependência, indicando as interfaces que um componente demanda de outros componentes. Esse mecanismo de representação de dependências torna o componente uma unidade encapsulada, de forma que o componente pode ser tratado de maneira independente. Como resultado disso, componentes e subsistemas podem ser reutilizados de maneira bastante flexível, sendo substituídos por meio da conexão de diversos componentes por meio de suas interfaces e dependências.

A linguagem UML suporte a especificação tanto de **componentes lógicos** (e.g. componentes de negócios, componentes de processo) como de **componentes físicos** (e.g. componentes EJB, componentes CORBA, componentes COM+ ou .NET, componentes WSDL, etc), junto com os artefatos que os implementam e os nós em que esses componentes são instalados e executados.

Um componente possui um caráter hierárquico, que pode ser explorado no processo de construção de um sistema. Desta forma, um componente pode ser visto, do ponto de vista externo, como um elemento executável do sistema, que se conecta com outros componentes para integrar a composição deste sistema. Por outro lado, de uma perspectiva interna, um componente pode ser visto como também um conjunto integrado de componentes menores que se integram, constituindo o componente em seu nível superior. Dessa forma, um componente representa uma parte modular de um sistema que encapsula seu conteúdo e cuja manifestação pode ser substituído no ambiente em que se insere.

Um componente define seu comportamento por meio da definição de suas interfaces disponibilizadas e das interfaces de que depende. Dessa forma, um componente pode ser substituído por outro componente que possua um mesmo conjunto de interfaces disponibilizadas. A construção de um sistema envolve a montagem de componentes, uns aos outros, por meio de suas interfaces. Um arranjo de componentes montado desta maneira, constitui-se de um **artefato**. Artefatos podem

ser instalados em diferentes ambientes de execução e colocados em execução nestes.

Um componente é uma unidade auto-contida que encapsula o estado e o comportamento de um grande número de objetos. Um componente especifica um contrato formal dos serviços que provê a seus clientes e dos serviços que necessita de outros componentes em termos de suas interfaces disponibilizadas e requeridas.

Um componente é uma unidade substituível que pode ser remanejada tanto durante o design como na implementação, por outro componente que lhe seja funcionalmente equivalente, baseado na compatibilidade entre suas interfaces. De modo similar, um sistema pode ser estendido adicionando-se novos componentes que tragam novas funcionalidades. As interfaces disponibilizadas e requeridas podem ser organizadas opcionalmente por meio de portas. Portas definem um conjunto de interfaces disponibilizadas e requeridas que são encapsuladas de maneira conjunta.

Certo número de estereótipos padrão são previstos para serem utilizados com componentes. Por exemplo, o estereótipo «subsystem» pode ser utilizado na modelagem de componentes de larga-escala. Os estereótipos «specification» e «realization» podem ser utilizados para representar componentes com especificações e realizações distintas, onde uma única especificação pode ter múltiplas realizações.

A notação UML para um componente segue a notação de uma classe estereotipada com o estereótipo «component». Opcionalmente, no canto superior direito, pode-se incluir um ícone específico, constituído por um retângulo maior e dois pequenos retângulos menores sobressaindo-se em seu lado esquerdo. Essa representação pode ser visualizada na figura 1 a seguir.

Figura 1: Exemplo da Notação de um Componente, com sua Interface disponibilizada

Na figura 2, apresenta-se um componente com suas interfaces disponibilizadas e requeridas. Neste exemplo, o componente Order disponibiliza as interfaces ItemAllocation e Tracking, e requer as interfaces Person, Invoice e OrderableItem.

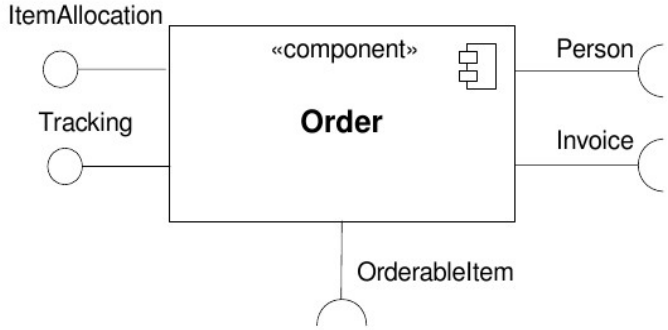

Figura 2: Exemplo da Notação de um Componente com Interfaces Disponibilizadas e Requeridas

A figura 3 a seguir apresenta outras notações para componentes, com diferentes compartimentos modelando diferentes aspectos do componente. Em ambos os casos, as interfaces dispobilizadas e requeridas estão representadas internamente em compartimentos, ao invés de externamente, por meio da conexão com interfaces.

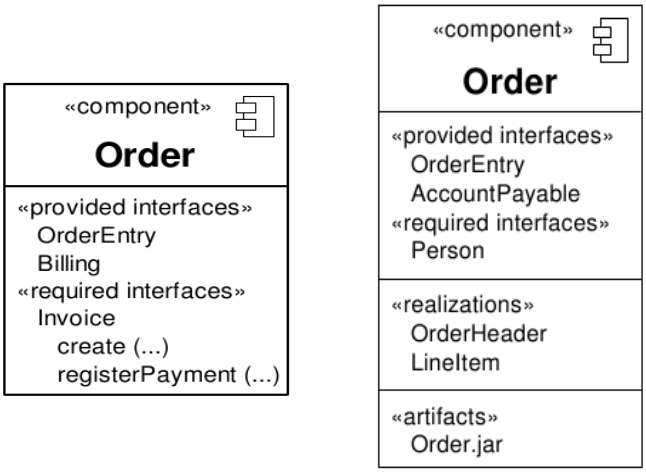

Figura 3: Outros Exemplos de Notação para Componentes

A figura 4 apresenta uma outra notação alternativa, onde as interfaces disponibilizadas e requeridas são modeladas detalhadamente.

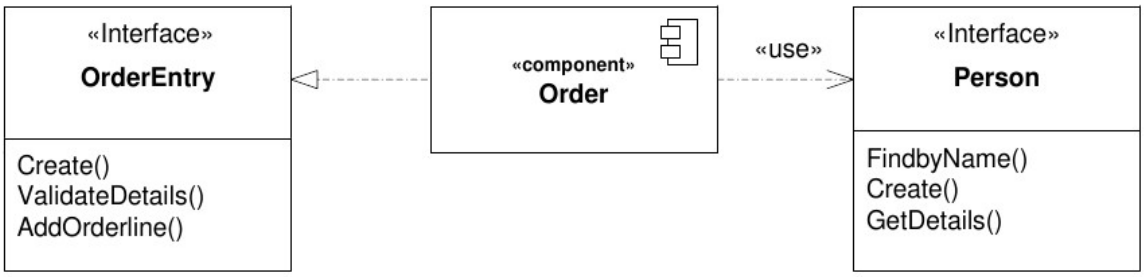

Figura 4: Notação para Componentes com Interfaces Detalhadas

A figura 5 a seguir, apresenta um exemplo em que as classes utilizadas para implementar um componente são mostradas de maneira explícita.

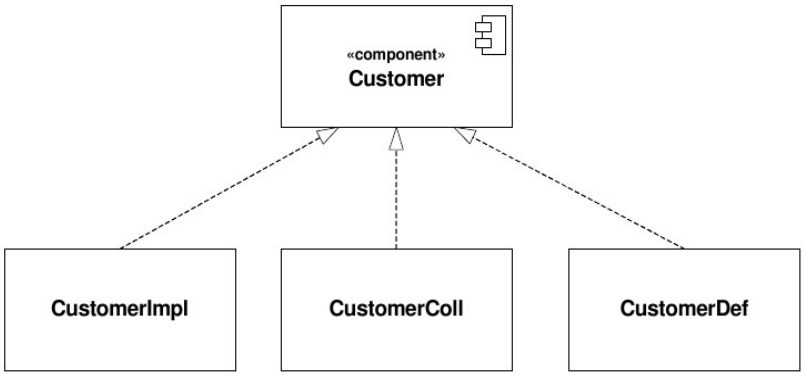

Figura 5: Classes que realizam um Componente

A figura 6 apresenta a estrutura interna de um componente, realizado pela composição de diversos outros componentes. Observe como as interfaces requeridas de alguns componentes são interligadas às interfaces disponibilizadas por outros componentes, para a montagem do componente de nível superior.

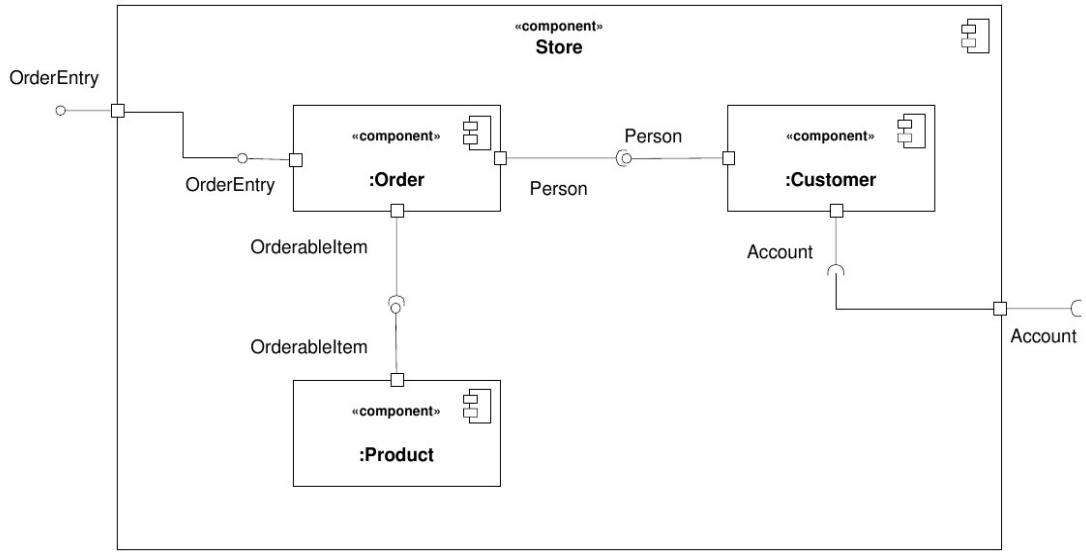

Figura 6: Perspectiva Interna de um Componente

Observe também na figura 6 acima, a definição de portas, representadas como pequenos quadrados onde as interfaces requeridas e disponibilizadas são conectadas aos componentes. Uma visão alternativa representando a dependência do componente **Order** dos componentes **Account** e **Product** pode ser visualizada na figura 7 a seguir:

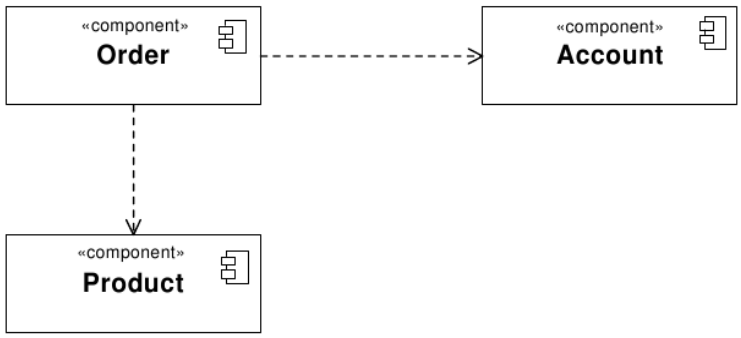

Figura 7: Representação alternativa para a dependência entre componentes

Uma outra representação alternativa pode ser vista ainda na figura 8.

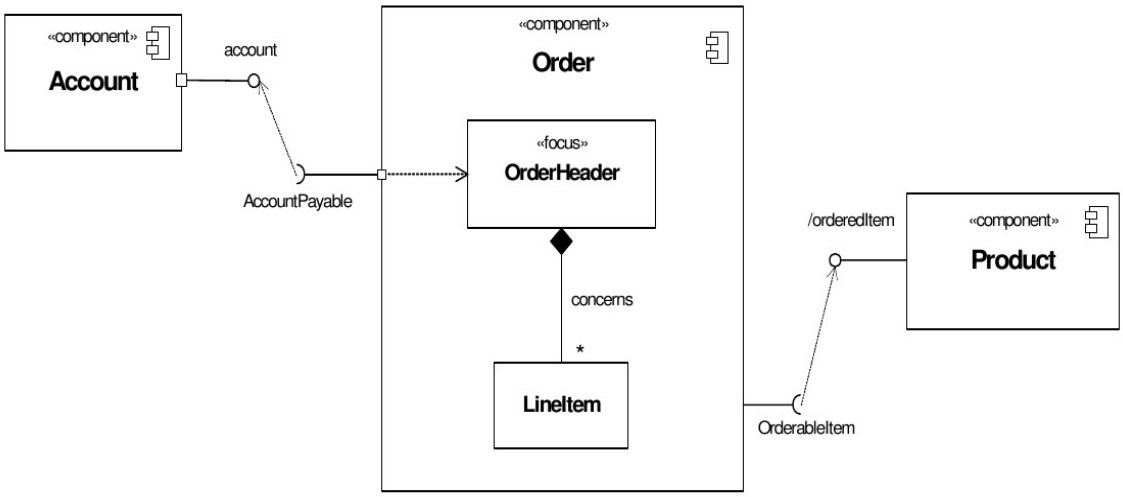

Figura 8: Outra visão alternativa para a montagem de componentes

Diagrama de Componentes

Diagramas de componentes, interconectando diferentes componentes em arranjos mais complexos, podem ser desenvolvidos conectando-se as interfaces disponibilizadas por um componente com as interfaces requeridas de outros componentes. Um exemplo dessa interconexão é apresentado na figura 9 a seguir:

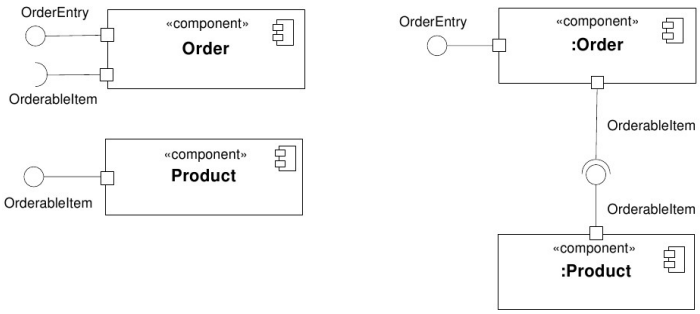

Figura 9: Composição de Sistemas pela Integração de Componentes

Em casos em que diversos componentes requerem a mesma interface, estes podem ser interconectados entre si, para efeito de simplificação. Um exemplo é apresentado na figura 10 a seguir:

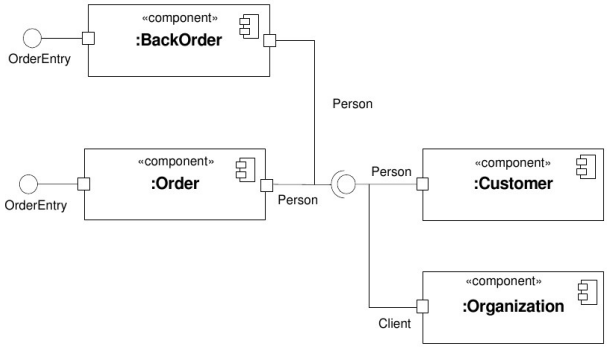

Figura 10: Interconexão de Interfaces Requeridas

Um exemplo de um diagrama de componentes, integrando diversos componentes em um artefato é mostrado na figura 11.

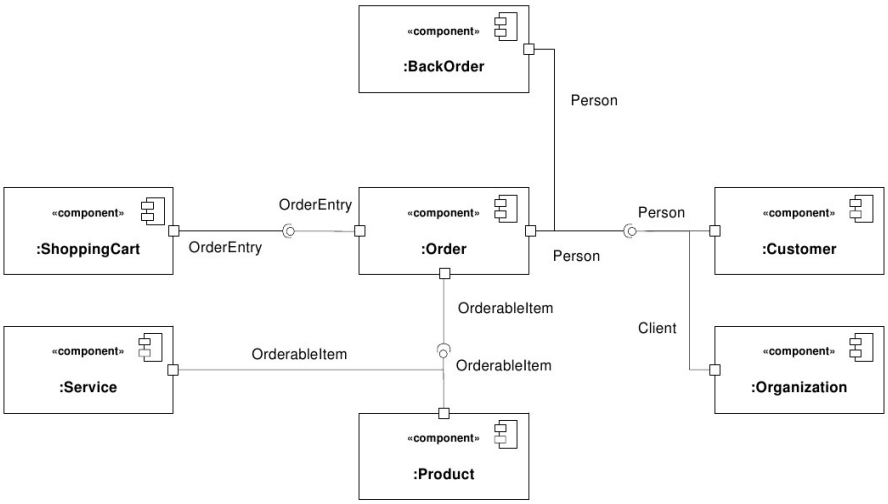

Figura 11: Exemplo de Diagrama de Componentes

Diagramas de Deployment

A linguagem UML prevê os assim chamados **diagramas de deployment** para representar uma estrutura física (normalmente de hardware), onde um conjunto de artefatos de software são instalados para compor uma configuração de um sistema.

Essa estrutura física é constituída por nós, conectados por vias de comunicação, criando uma rede de complexidade arbitrária. Nós são tipicamente definidos de maneira recursiva, podendo representar tanto dispositivos de hardware como ambientes de execução de software. Artefatos representam elementos concretos do mundo físico, resultados de um processo de desenvolvimento. Os diagramas de deployment normalmente são suficientes para representar a grande maioria das aplicações modernas de sistemas computadorizados. Em casos onde modelos de instalação mais elaborados sejam necessários, os diagramas de deployment podem ser estendidos por meio de profiles ou meta-modelos, de forma a representar os ambientes de hardware ou software desejados.

Artefatos

Um **artefato** é a especificação de um conjunto concreto de informações que é utilizado ou produzido por um processo de desenvolvimento de software, ou para a instalação ou operação de um sistema computacional. Exemplos de artefatos incluem arquivos de modelos, arquivos de código-fonte, scripts, arquivos executáveis, tabelas em bancos de dados, documento de texto, mensagem de e-mail ou qualquer outro resultado de um processo de desenvolvimento.

Um artefato representa, portanto, um elemento concreto do mundo físico. Uma instância particular do artefato (ou cópia do artefato) é instalada em uma instância de um nó. Artefatos podem manter associações com outros artefatos que podem estar aninhados dentro de si próprio. Diversos estereótipos estão previstos na norma, especificando o tipo de artefato. Por exemplo, «source» ou «executable». Estes estereótipos podem ser ainda especializados mais ainda em um profile. Por exemplo, o estereótipo «jar» poderia ser definido como uma subclasse de «executable» de forma a designar arquivos Java executáveis.

Na linguagem UML, um artefato é apresentado utilizando-se o retângulo de uma classe ordinária, com o uso da palavra-chave «artifact». Alternativamente, este retângulo pode acrescentar ainda um pequeno ícone no canto superior direito do retângulo.

Um exemplo de um artefato pode ser visualizado na figura 12 a seguir:

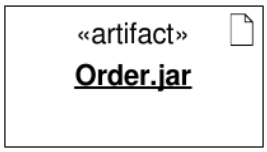

Figura 12: Exemplo de um Artefato

Na figura 13, apresenta-se um exemplo mostrando como pode-se detalhar a constituição de um artefato (por meio de componentes), utilizando-se de relações de dependência estereotipadas.

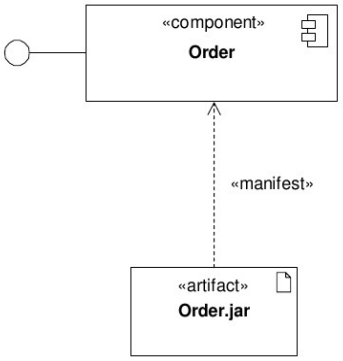

Figura 13: Exemplo Mostrando a Constituição de um Artefato

Nós

Um **nó** é um recurso computacional onde artefatos podem ser instalados para posterior execução. Nós podem ser interconectados por meio de conexões que definem estruturas de redes. Estas conexões representam caminhos de comunicação possíveis entre os nós. Topologias específicas de redes podem ser definidas por meio dessas conexões. Nós hierárquicos (ou seja, nós dentro de nós) podem ser definidos utilizando-se associações do tipo composição, ou definindo-se uma estrutura interna para aplicações de modelagem avançada.

Arcos tracejados com o uso do keyword «deploy» podem ser utilizados para representar a capacidade de um nó de suportar um determinado tipo de artefato. Alternativamente, isso pode ser representado utilizando-se artefatos aninhados dentro do nó. Em ambos os casos, isso significa que o artefato correspondente encontra-se instalado no nó.

A figura 13 a seguir, ilustra a notação de um nó.

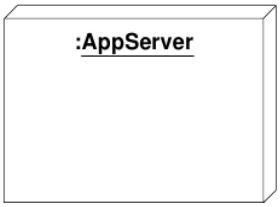

Figura 13: Exemplo da Notação de um Nó

A figura 14 apresenta dois artefatos instalados em um nó.

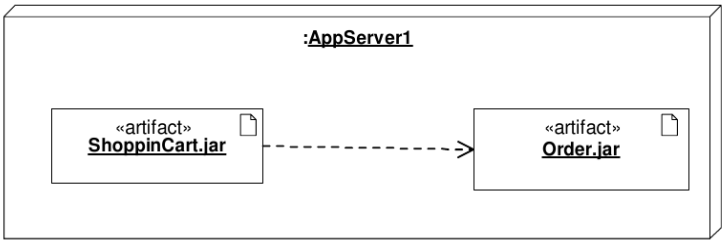

Figura 14: Exemplo de Artefatos instalados em um nó

A figura 15 ilustra uma notação alternativa para indicar a instalação destes mesmos artefatos no nó.

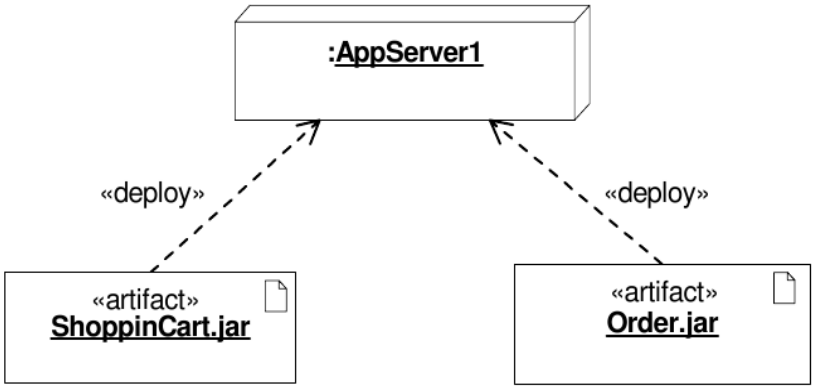

Figura 15: Notação alternativa para indicar a instalação de artefatos em um nó

A figura 16 a seguir ilustra ainda uma terceira maneira de se representar a instalação de artefatos em um nó.

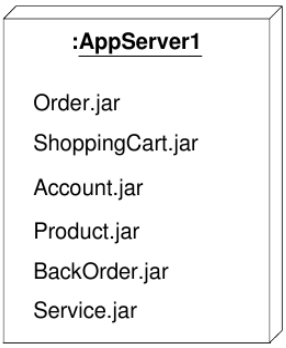

Figura 16: Outra alternativa para representar a instalação de artefatos em um nó

A figura 17 mostra a instalação de artefatos aninhados em um nó.

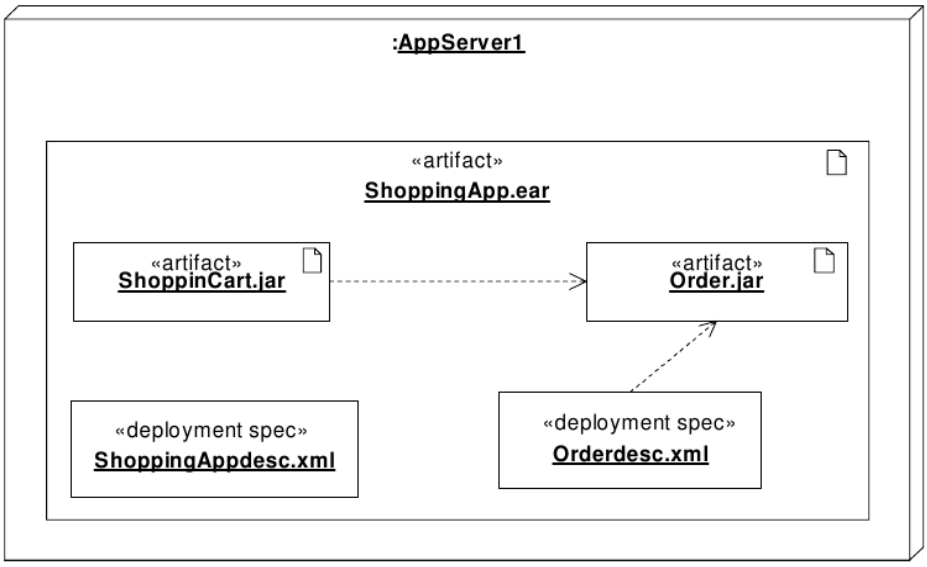

Figura 17: Exemplo de Artefatos aninhados instalados em um nó

A figura 18 mostra dois nós conectados entre si.

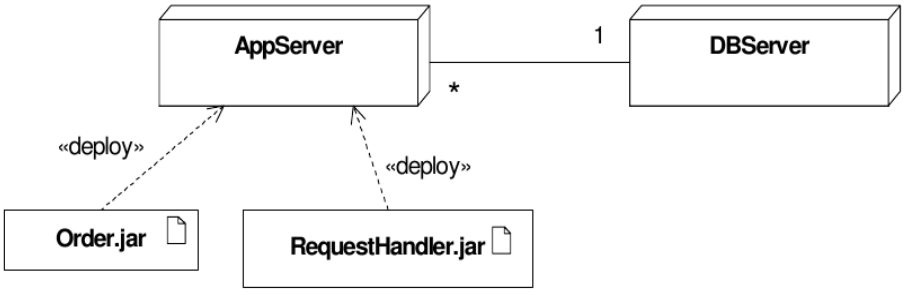

Figura 18: Exemplo de Conexão entre Nós

Nós podem ainda receber estereótipos, para representar diferentes tipos de plataformas de execução. Exemplos de estereótipos desse tipo incluem os estereótipos «device» para representar plataformas de hardware e «executionEnvironment» para representar plataformas de software com ambientes executáveis. Um exemplo de um nó estereotipado como «executionEnvironment» pode ser visto na figura 19.

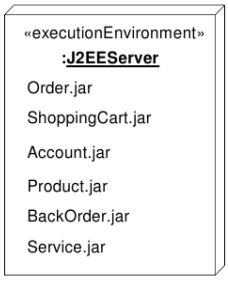

Figura 19: Exemplo de Nó estereotipado

Na figura 20, vemos como nós estereotipados podem ser aninhados um sobre os outros para representar sistemas mais complexos.

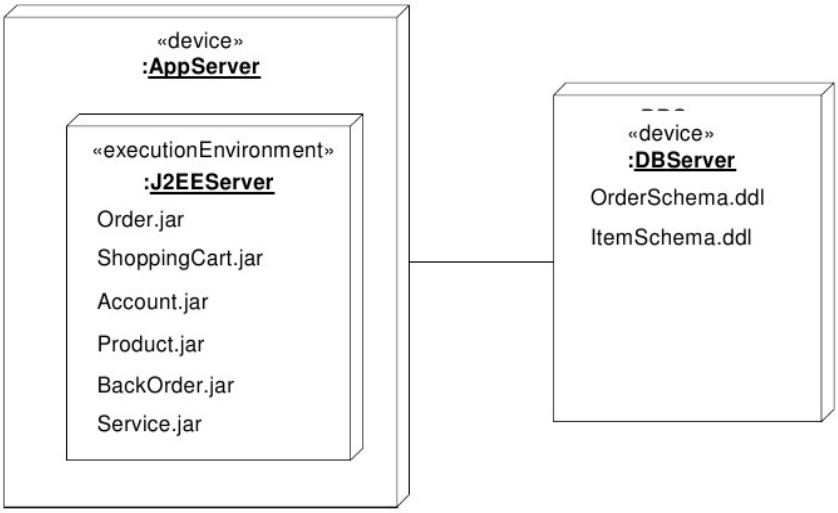

Figura 20: Exemplo de aninhamento de Nós estereotipados