# Mô hình hóa SOA: Phần 5. Cài đặt dịch vụ

Mức độ: Trung bình

Jim Amsden, Chuyên viên kỹ thuật cao cấp, IBM, Software Group

06 07 2009

Trong bốn bài viết trước đây về chủ đề này (xem "Thông tin liên quan về chủ đề này"), chúng tôi đã sử dụng phân tích nghiệp vụ để phân biệt các dịch vụ liên quan đến nghiệp vụ ("Mô hình hóa SOA: Phần 1. Xác định dịch vụ") đáp ứng được những mục đích và mục tiêu của nghiệp vụ. Sau đó chúng ta sẽ tìm hiểu rõ các dịch vụ và sử dụng các giao thức ("Mô hình hóa SOA: Phần 2. Đặc tả dịch vụ") cần thiết để đáp ứng những mục tiêu IT. Tiếp theo, chúng tôi đã cung cấp những mô hình thiết kế để nhận biết những dịch vụ được cung cấp và sử dụng như thế nào ("Mô hình hóa SOA: Phần 3. Thực hiện dịch vụ," "Mô hình hóa SOA: Phần 4. Các thành phần tạo nên dịch vụ"). Kết quả là một công nghệ trung gian nhưng là một mô hình thiết kế hoàn chỉnh của một giải pháp các dịch vụ được kiến trúc hóa. Trong bài viết cuối cùng về chủ đề này, chúng ta xem cách tạo ra một cài đặt hiện thời phù hợp với những quyết định về kiến trúc và thiết kế được đề cập trong mô hình các dịch vụ. Chúng ta sẽ tạo ra cài đặt dựa trên nền tảng bằng cách khai thác cả hai mô hình định hướng phát triển và công cụ chuyển đổi IBM® Rational® Software Architect UML-to-SOA để tạo ra một dịch vụ Web từ mô hình SOA.

## Nội dung của bài viết này

Các bước trong những bài viết trước đây đã tạo ra một mô hình giải pháp SOA hoàn chỉnh để đáp ứng những yêu cầu về nghiệp vụ. Bởi vậy, chúng ta biết những yêu cầu nào kiến trúc giải pháp này cần đáp ứng và cần phải thay đổi như thế nào khi yêu cầu thay đổi.

Để triển khai và chạy một dịch vụ Web, chúng ta cần tạo một cài đặt hiện thời phù hợp với những quyết định về kiến trúc và thiết kế được đề cập trong mô hình. Chúng ta có thể tạo ra giải pháp đó theo cách thủ công, sử dụng mô hình như một chỉ dẫn. Nhưng cách này rất dài dòng, dễ xảy ra lỗi, và tốn thời gian, và nó yêu cầu một người phát triển có trình độ cao để đảm bảo rằng những quyết định kiến trúc đã được cài đặt một cách chính xác. Hoàn toàn có thể tạo ra giải pháp bằng cách thủ công, và sử dụng mô hình như một hướng dẫn sẽ rất hữu ích. Nhưng sử dụng một mô hình hoàn thiện, chính thức và đã được kiểm chứng giúp chúng ta có thể để khai thác mô hình định hướng phát triển (MDD) để tạo ra một hay nhiều khung giải pháp từ mô hình và sau đó hoàn thiện mã hóa chi tiết trong môi trường lập trình dựa trên nền tảng. Đó là chủ đề chính của bài viết này. Chúng ta sẽ sử dụng công cụ chuyển đổi IBM® Rational® Software Architect UML-to-SOA để tạo ra giải pháp dịch vụ Web mà có thể được sử dụng một cách trực tiếp trong IBM® WebSphere® Integration Developer để cài đặt, kiểm tra, và triển khai một giải pháp đã được hoàn thiện.

Cũng như tất các các bài viết khác về chủ đề này, chúng tôi sẽ sử dụng công cụ Rational và WebSphere để xây dựng những công cụ giải pháp và ghép nối chúng với nhau, từ đó chúng ta có thể kiểm tra với những yêu cầu đã đề ra và quản lý thay đổi hiệu quả hơn. Bảng 1 cung cấp tóm tắt về quá trình tổng quan mà chúng ta sẽ sử dụng để phát triển các ví dụ và các công cụ được sử dụng để xây dựng các giải pháp.

**Bảng 1. Những vai trò, nhiệm vụ và công cụ của quá trình phát triển**

| Vai trò | Nhiệm vụ | Công cụ |
|---|---|---|
| Giám đốc Nghiệp vụ | Đề ra mục đích và mục tiêu nghiệp vụ | IBM® Rational® RequisitePro® |
| Nhà phân tích Nghiệp vụ | Phân tích những yêu cầu nghiệp vụ | IBM® WebSphere® Business Modeler |
| Kiến trúc sư Phần mềm | Thiết kế kiến trúc của giải pháp | IBM Rational Software Architect |
| | | |

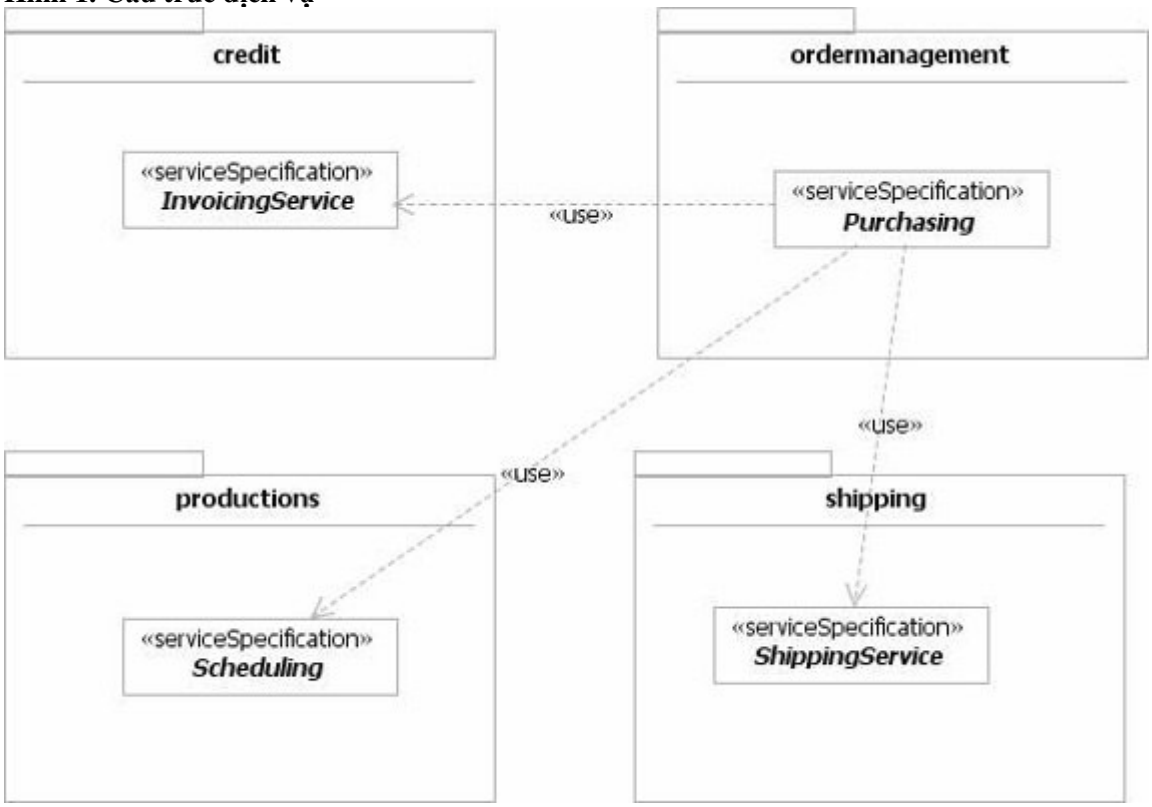
| | | |
|---|---|---|
| Người phát triển dịch vụ Web | Cài đặt giải pháp | IBM® Rational® Application Developer và IBM WebSphere Integration Developer |

Nhưng trước khi bắt đầu, chúng ta hãy xem lại các dịch vụ và đối tượng tham gia dịch vụ mà chúng ta đã tạo ra trong các bài viết trước đây.

# Xem lại đặc tả, điều khoản, và sử dụng của dịch vụ

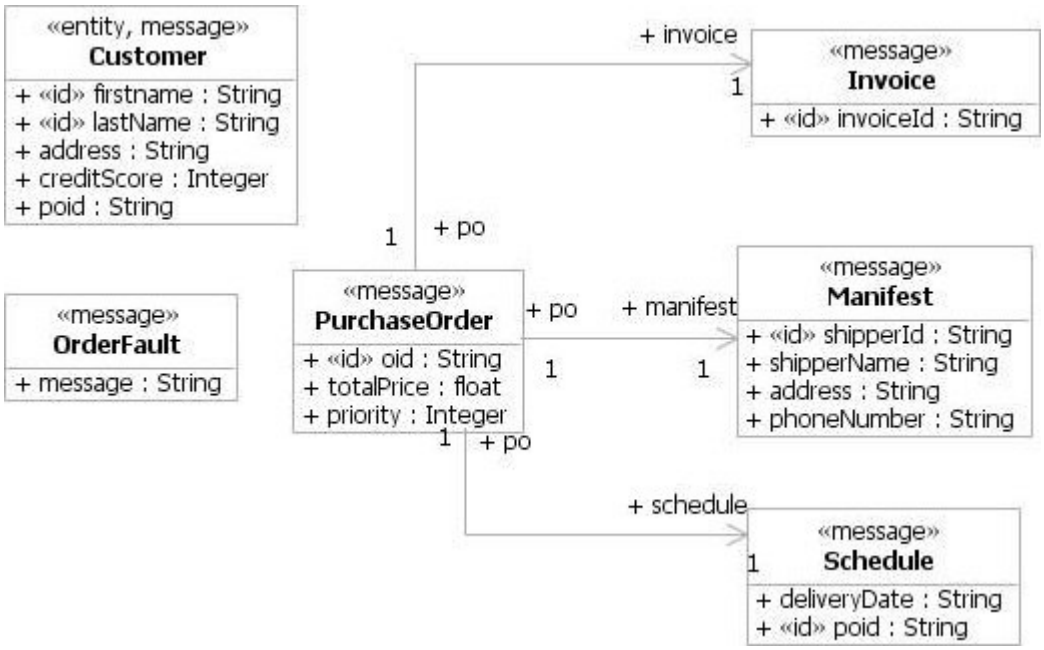
Hình 1 biểu thị những đặc điểm dịch vụ được đưa ra để đáp ứng những yêu cầu nghiệp vụ. Đây là những dịch vụ có khả năng đóng các vai trò trong hợp đồng Business Services Requirements này.

**Hình 1. Cấu trúc dịch vụ**




Hình 2 biểu thị mô hình dữ liệu cho dịch vụ dữ liệu. Đây là mô hình của thông tin được trao đổi giữa những khác hàng và các nhà cung cấp dịch vụ, và nó được sử dụng để định nghĩa các tham số của các hoạt động dịch vụ.

**Hình 2. Mô hình dữ liệu dịch vụ**



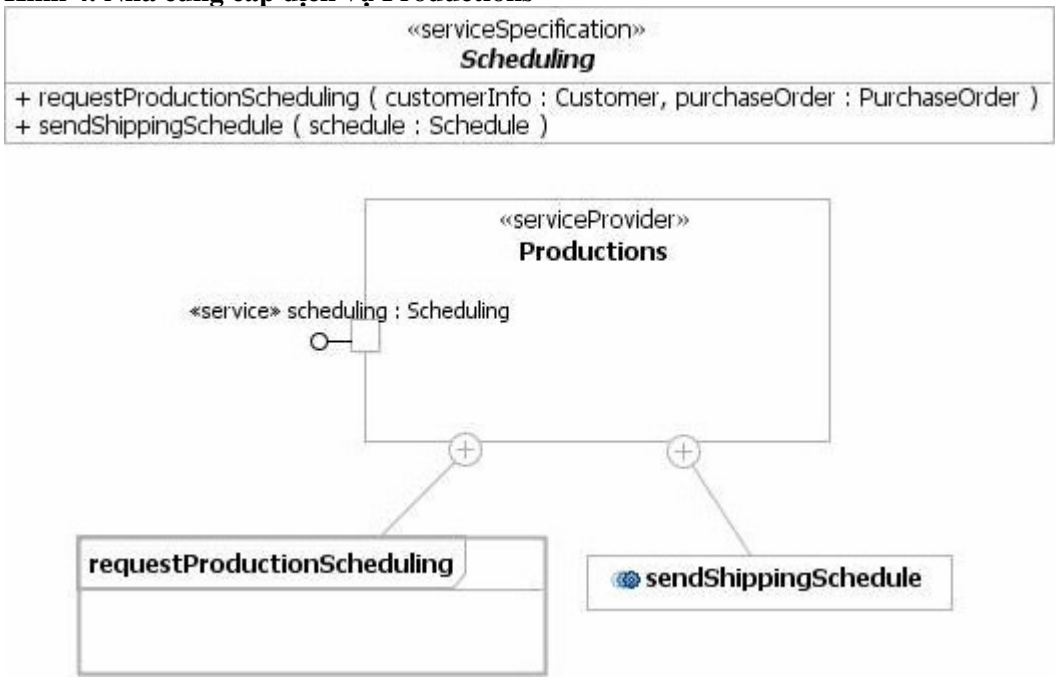

## Lập lịch (Scheduling)

Hình 3 biểu thị đặc điểm dịch vụ Lập lịch (Scheduling) hoàn chỉnh.

**Hình 3. Đặc điểm dịch vụ Lập lịch (Scheduling)**


Đặc điểm dịch vụ này là một giao diện đơn giản, bởi vì nó không có giao thức để sử dụng những dịch vụ lập biểu. Hình 4 biểu thị một nhà cung cấp dịch vụ cung cấp dịch vụ Lập lịch.

**Hình 4. Nhà cung cấp dịch vụ Productions**



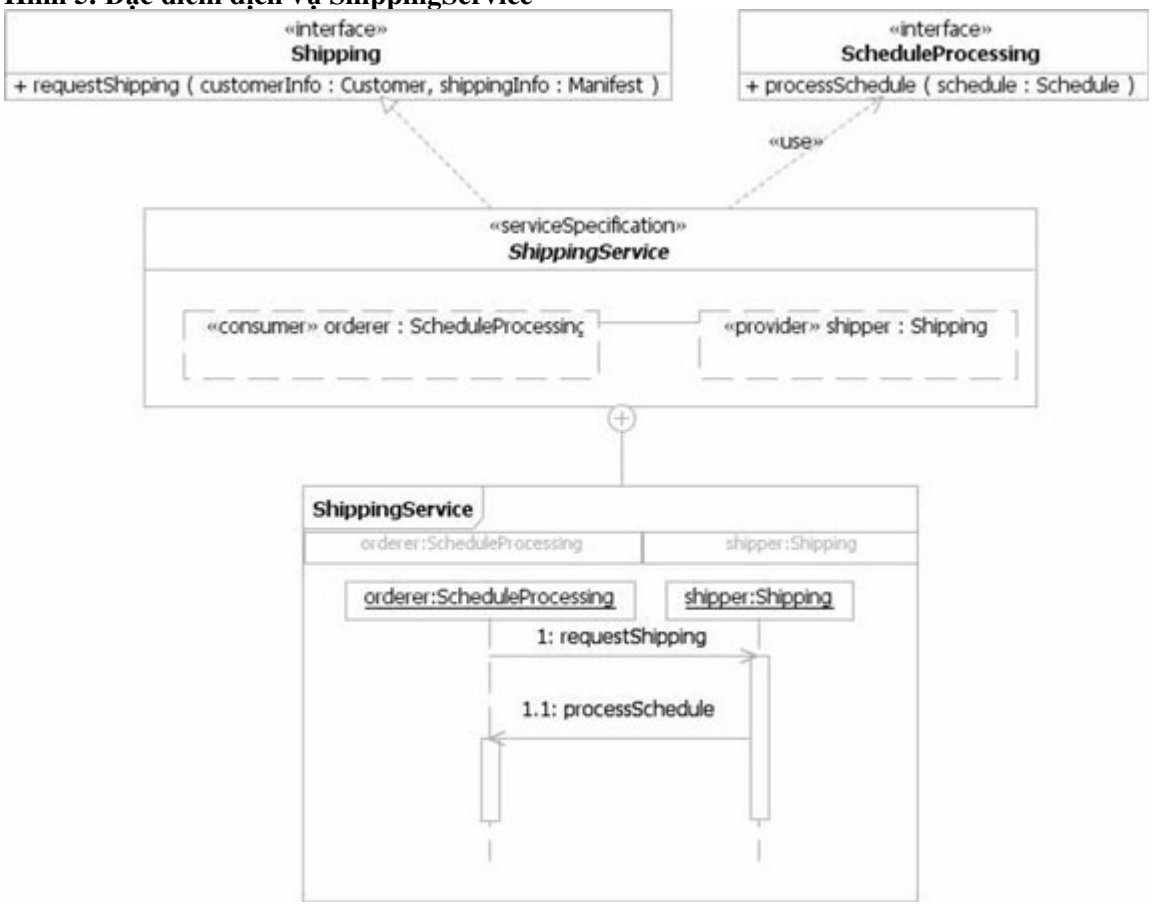

Những cài đặt hiện thời của các cách xử lý requestProductionScheduling và sendShippingSchedule không

được hiển thị chi tiết trong sơ đồ này, nhưng chúng được định nghĩa trong mô hình.

### Vận chuyển (Shipping)

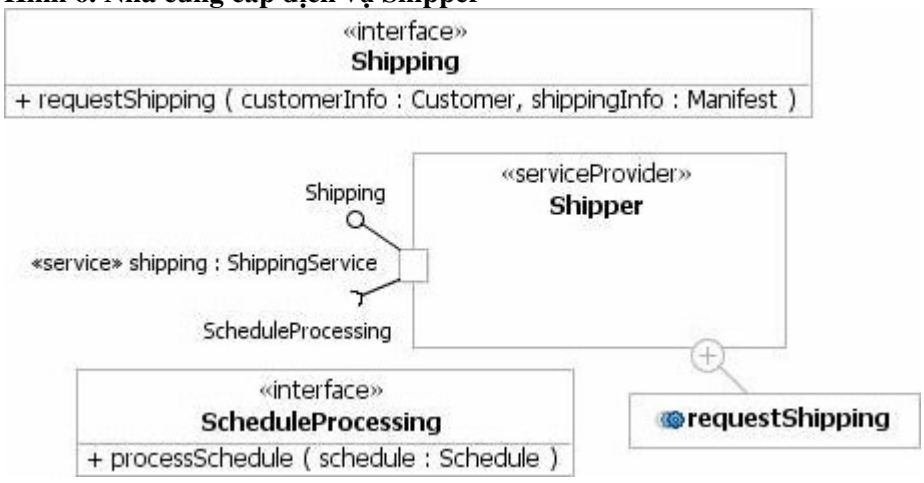
Hình 5 biểu thị đặc điểm dịch vụ ShippingService.

**Hình 5. Đặc điểm dịch vụ ShippingService**




Đặc điểm dịch vụ này phức tạp hơn một chút, bởi vì nó mô hình hóa sự tương tác giữa một người vận chuyển hàng và một người đặt hàng bằng tương tác gọi ngược đơn giản. Hình 6 biểu thị nhà cung cấp dịch vụ ShippingService.

**Hình 6. Nhà cung cấp dịch vụ Shipper**



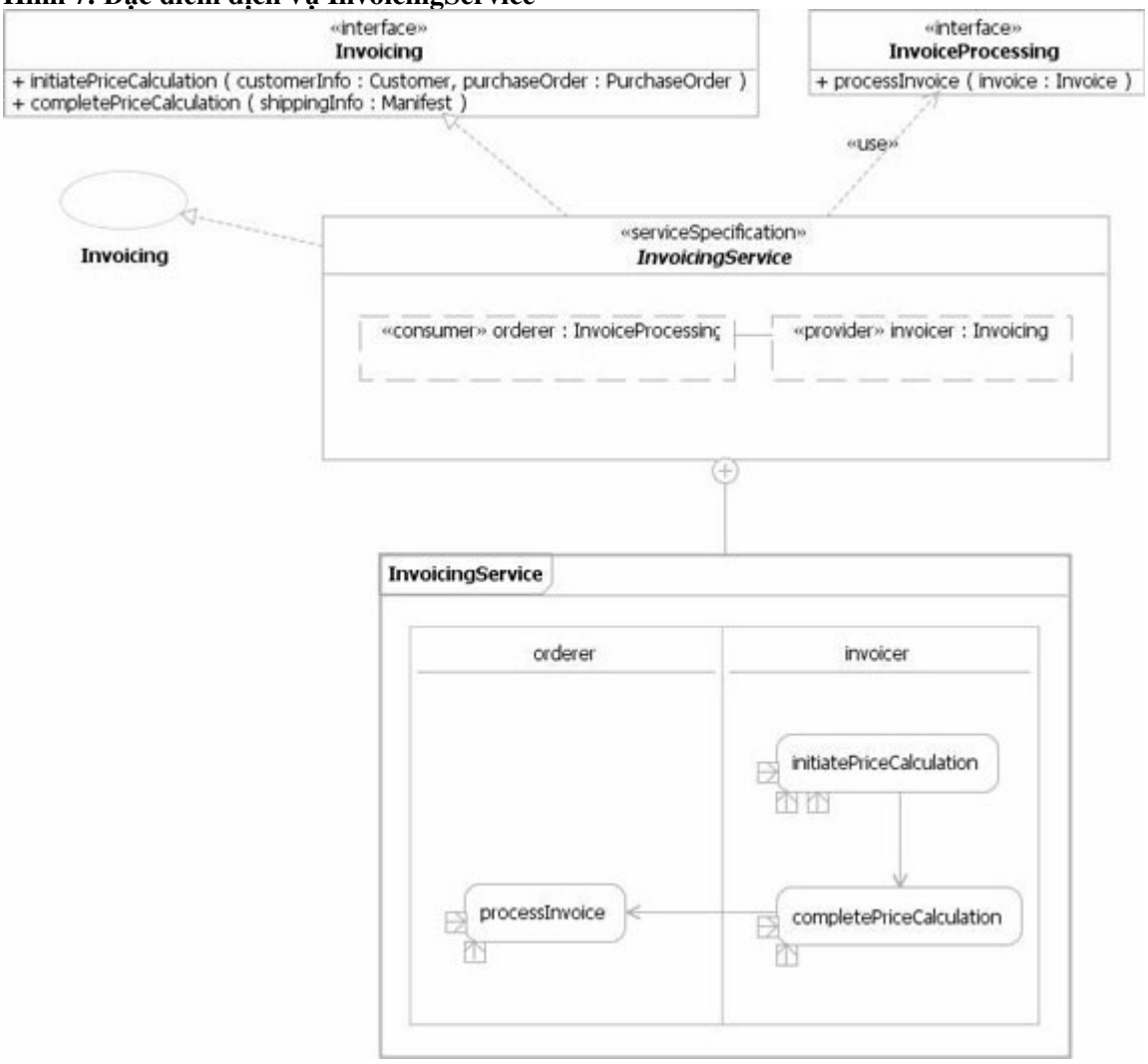

Các xử lý mờ requestShipping là phương pháp cho hoạt động requestShipping được cung cấp để gọi ra processSchedule trên cảng vận chuyển hàng đi trong cài đặt của nó. Khách hàng liên hệ với cảng này sẽ

được yêu cầu cung cấp một cài đặt của hoạt động này để phản hồi các yêu cầu.

### Lập hóa đơn (Invoicing)

Hình 7 biểu thị đặc điểm dịch vụ InvoicingService.

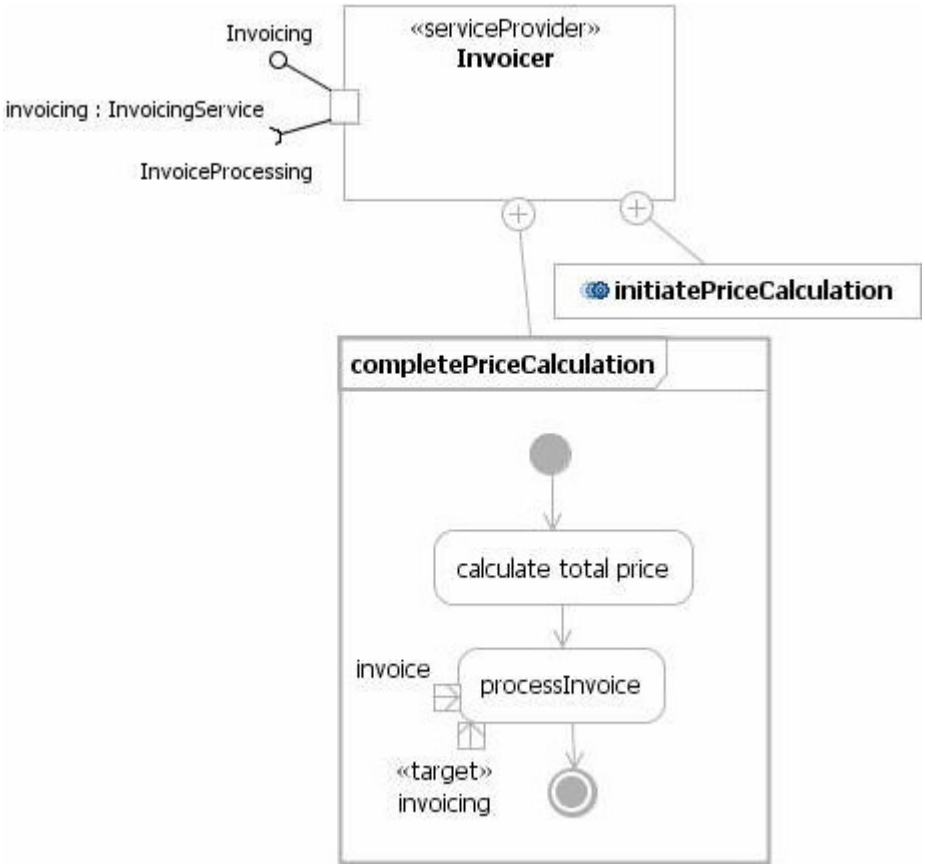
**Hình 7. Đặc điểm dịch vụ InvoicingService**




Giao thức này phức tạp hơn một chút. Nó chỉ ra rằng các khả năng dịch vụ phải được sử dụng trong một đơn đặt hàng cụ thể. Cả khách hàng và nhà cung cấp đều được yêu cầu tuân theo giao thức này. Trong trường hợp này, một hoạt động được sử dụng để định nghĩa giao thức dịch vụ.

Hình 8 biểu thị nhà cung cấp dịch vụ InvoicingService và những phương pháp cung cấp các khả năng dịch vụ.

**Hình 8. Những cài đặt dịch vụ Invoicer**




## Mua hàng (Purchasing)

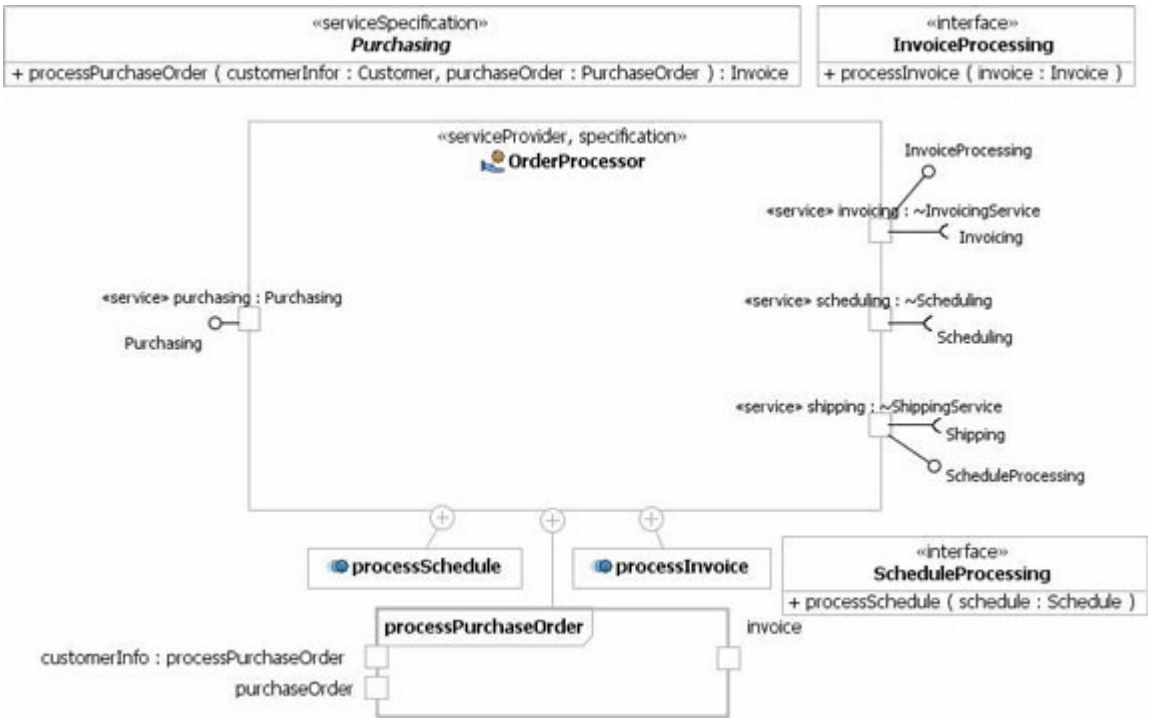
Cuối cùng, Hình 9 biểu thị đặc điểm dịch vụ Mua hàng (Purchasing).

**Hình 9. Đặc điểm dịch vụ Mua hàng Purchasing**


Đặc điểm dịch vụ này phản ánh khả năng chức năng được miêu tả trong quy trình nghiệp vụ Process Purchase Order đầu tiên. Nó phản ánh một dịch vụ được định nghĩa và thiết kế từ quy trình nghiệp vụ.

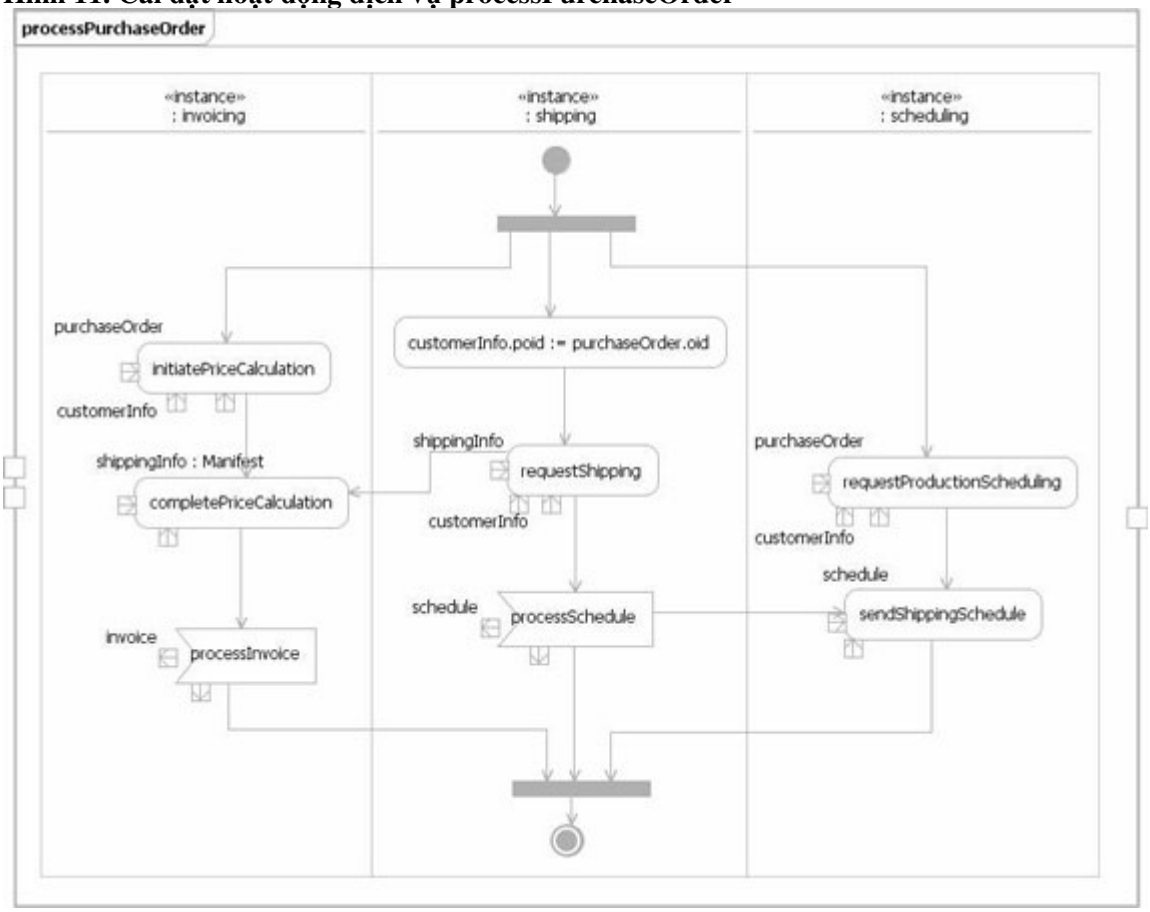
Hình 10 biểu thị một nhà cung cấp dịch vụ Mua hàng, và các dịch vụ mà nó yêu cầu thực hiện.

**Hình 10. Nhà cung cấp dịch vụ OrderProcessor**




Hình 11 biểu thị phương pháp hoàn thiện dành cho hoạt động dịch vụ processPurchaseOrder sử dụng một hoạt động UML.

**Hình 11. Cài đặt hoạt động dịch vụ processPurchaseOrder**




Sơ đồ này gần giống với sơ đồ IBM WebSphere Business Modeler dành cho cùng một cách xử lý. Những hoạt động dịch vụ InvoiceProcessing và ScheduleProcessing được nhận ra qua các hành động *Accept Event* processInvoice và processSchedule trong quá trình xử lý. Chú ý rằng các hoạt động tương ứng trong

những giao diện được biểu thị như các hoạt động `<<trigger>>` để chỉ ra khả năng phản hồi với AcceptCallActions (giống như những tiếp nhận và các AcceptEventActions mà ở đó trigger là một SignalEvent). Từ khóa <<trigger>> không phải là một phần của Ngôn ngữ Mô hình hóa Hợp nhất 2 (Unified Modeling Language 2, viết tắt thành UML 2); nó chỉ bao gồm để làm nổi bật những hoạt động này được nhận ra như thế nào. Chú ý rằng những hoạt động này sẽ không được chấp nhận trừ phi hoạt động processPurchaseOrder đang chạy và luồng của sự kiểm soát đạt được hai hành động Accept Call. Điều này chỉ ra rằng sự cài đặt của một hoạt động có thể quyết định thời điểm đưa ra các phản hồi cho những hoạt động khác.

# Những ứng dụng điển hình của mô hình hóa các dịch vụ

Mô hình hóa UML 2 giúp chúng ta hiểu sâu hơn về các hệ thống tầng dưới. Tuy nhiên, mô hình hóa không phải là một công cụ đa năng, và những sơ đồ mô hình phức tạp vẫn có thể phải cần những người am hiểu để tạo ra và hiểu được. Đây là một hệ quả tự nhiên của sự cần thiết để hỗ trợ một mô hình chung của sự tính toán có thể được sử dụng rộng rãi cho các trình miền ứng dụng và các mức độ của trừu tượng, và nó cũng phản ánh những ngữ nghĩa học của các mô hình thực hiện dựa trên nền tảng. Thêm nữa, những loại của hành động hay kiểu của các mô hình hoạt động có thể cần được ràng buộc để cho phép sự chuyển đổi của những mô hình đó sang nền tảng thực hiện mục tiêu một cách hiệu quả hơn.

Trong trường hợp này, nền tảng mục tiêu là những dịch vụ Web và Mô hình lập trình IBM SOA được hỗ trợ bởi WebSphere Integration Developer. Nền tảng này bao gồm những đối tượng nghiệp vụ (XSD), giao diện (WSDL), bộ phận mô đun (SCDL, một tiền thân IBM của SCA), các quy trình (BPEL), SCDL (máy trạng thái nghiệp vụ), và những thành phần Java™. Để hỗ trợ việc chuyển đổi từ những mô hình UML thành nền tảng dịch vụ Web cụ thể này, chúng ta cần làm theo những ví dụ về mô hình dịch vụ. Đầu tiên chúng ta cần tùy biến UML để mô hình hóa một kiến trúc giải pháp SOA, và sau đó chúng ta cần sử dụng một kiểu mô hình cụ thể để tạo ra những mô hình mà có thể dịch sang những dịch vụ Web.

UML được tùy biến đặc biệt bằng cách sử dụng các hồ sơ. Một **hồ sơ** định nghĩa một số lớp mẫu có thể mở rộng thành một hoặc nhiều siêu lớp UML với thêm nhiều đặc tính và mối quan hệ. Những hồ sơ được áp dụng đối với các mô hình UML để kích hoạt những đuôi mở rộng này. Những hồ sơ được áp dụng này thường hỗ trợ hai mục đích trong các kiến thức hướng mô hình:

- Mục đích đầu tiên của những hồ sơ là để tùy biến những trừu tượng mà có xu hướng được hỗ trợ. Trong trường hợp này, chúng ta đã áp dụng hồ sơ IBM Software Services cho mô hình Purchase Order Process để hỗ trợ mô hình những dịch vụ bằng việc mở rộng UML. Rất nhiều lớp mẫu trong hồ sơ đó giải thích cách các siêu lớp UML được sử dụng để mô hình hóa một giải pháp SOA và để hạn chế một số thứ có thể đã được hoàn thành trong UML nhằm đảm bảo rằng các mô hình SOA phản ánh các nguyên tắc SOA. Ví dụ, lớp mẫu `<<serviceConsumer>>` được sử dụng để biểu thị một thành phần UML đang mô hình hóa một tác nhân của các dịch vụ không hoàn toàn cung cấp bất cứ dịch vụ tái sử dụng của riêng nó. Một thành phần như vậy có thể miêu tả một quy trình nghiệp vụ không được tái sử dụng. Ví dụ về sự giới hạn, hồ sơ Software Services yêu cầu tất cả giao diện được nhận biết và sử dụng bởi một thành phần `<<serviceProvider>>` để được xử lý thông qua các cổng dịch vụ, một cách gián tiếp. Điều này nhằm để giảm kết nối giữa người dùng được kết nối và các nhà cung cấp bằng việc tạo ra các phụ thuộc trên cổng dịch vụ. Sau đó, khi một số giao diện thay đổi, chỉ cần kiểm tra sự phản hồi của cổng đó với sự thay đổi, mà không cần phải kiểm tra tất cả các cổng kết nối với thành phần.
- Mục đích thứ hai của những hồ sơ trong kiến trúc hướng mô hình (MDA) là để hỗ trợ những đánh dấu hướng nền tảng. Những đánh dấu này bao gồm những lớp mẫu và những thuộc tính thêm vào "đánh dấu" một mô hình với những thông tin cần thiết để dịch mô hình thành một nền tảng cụ thể. Ví dụ, một gói UML có thể cần có một Uniform Resource Identifier cụ thể (URI) khi được dịch thành một trình chứa các dịch vụ Web.

Trong một số trường hợp, hai mục đích của những hồ sơ này được gộp thành một. Ví dụ, những đuôi mở rộng để UML hỗ trợ mô hình hóa dữ liệu tương quan bao gồm một hồ sơ đơn lẻ sẽ mở rộng UML cho mô hình dữ liệu thực thể phụ thuộc tương đối (ERA) và cung cấp những đánh dấu cần thiết để dịch những mô hình miền UML sang những mô hình dữ liệu mang tính logic IBM® Rational® Data Architect (LDMs). Hồ sơ IBM Software Services cung cấp cả hai vai trò này cho những mô hình được dịch thành dịch vụ.

Trong những trường hợp khác, một hồ sơ có thể được sử dụng cho việc hỗ trợ mô hình hóa, trong khi một

hồ sơ khác được sử dụng để định hướng quá trình dịch. Ví dụ, coi trường hợp của những thiết kế mô hình SOA đã được cài đặt trong Java™ 2 Platform, Enterprise Edition (J2EE). Ứng dụng sau đó có thể được trợ giúp bằng việc áp dụng cả hồ sơ Software Services và hồ sơ chuyển đổi Enterprise Java™Beans (EJB) cho cùng một mô hình. Những lớp mẫu hồ sơ Software Services sẽ được áp dụng cho những yếu tố mô hình để hỗ trợ mô hình hóa những dịch vụ, trong khi những lớp mẫu hồ sơ chuyển đổi EJB sẽ được áp dụng cho những yếu tố mô hình để hướng dẫn việc thực hiện của công cụ chuyển đổi Rational Software Architect UML-to-EJB khi nó sinh ra mã trình cài đặt. Tất nhiên, Cùng một mô hình SOA đó cũng có thể được dịch sang những dịch vụ Web bằng việc sử dụng công cụ chuyển đổi Rational Software Architect UML-to-SOA. Công cụ chuyển đổi Rational Software Architect UML-to-SOA sẽ sinh ra những dịch vụ Web bằng cách mở khóa hồ sơ đánh dấu Software. Nó sẽ từ chối những đánh dấu cho công cụ chuyển đổi EJB.

Những phần tiếp theo miêu tả một số ví dụ mô hình hóa điển hình cho những mô hình SOA sẽ được dịch thành những dịch vụ Web, đặc biệt những dịch vụ Web được cài đặt trong IBM® SOA Programming Model và khi được hỗ trợ bởi công cụ mô hình hóa WebSphere Integration Developer.

## Mô hình hóa dữ liệu

Kiểu của một tham số thao tác dịch vụ nên là một kiểu nguyên gốc cũng như một DataType `<<message>>`. Những người tạo mô hình không nên tạo sự giả định về vị trí của dữ liệu, thuật ngữ tham trị hay tham chiếu, hay bất cứ tiện nghi quản lý đồng quy ẩn . Giả sử các cài đặt đang làm việc trên bản sao tạm thời của dữ liệu mà có thể được chuyển giao, chuyển đổi, hay cả hai từ nguồn ban đầu. Điều này đảm bảo sự liên kết dữ liệu tối thiểu giữa người sử dụng dịch vụ và nhà cung cấp dịch vụ.

## Mô hình hóa các dịch vụ

Như đã được đề cập trong hồ sơ IBM Software Services, một thành phần cung cấp dịch vụ phải gián tiếp nhận ra được hoặc sử dụng tất cả các giao diện thông qua các cổng dịch vụ. Điều này đảm bảo sự liên kết hợp lí giữa những khách hàng dịch vụ và các nhà cung cấp được kết nối với thành phần.

## Mô hình hóa hành động

Một nhà cung cấp dịch vụ cụ thể mô hình hóa các phương pháp cho các hoạt động của nó bằng cách cung cấp một phương thức xử lý cho mỗi hoạt động.

**Chú ý:**
Một nhà cung cấp dịch vụ **cụ thể** không phải là một thành phần trừu tượng và thành phần `<<specification>>` .

Bất kì phương thức xử lý nào cũng có thể được sử dụng, nhưng nếu nền tảng mục tiêu mô hình là những dịch vụ Web, nó phải đảm bảo sự thuận tiện trong việc sử dụng một hoạt động mà hoạt động đó có thể dễ dàng được dịch thành WebSphere-BPEL. Mô hình hoạt động trong Hình 11 là một cách xử lý riêng của thành phần nhà cung cấp dịch vụ Order Processor, nó là phương pháp cho hoạt động processPurchaseOrder. Có một số điểm về hoạt động này cần được giải thích sâu hơn:

- Chữ kí cho một phương pháp xử lý phải tương ứng với đặc điểm hoạt động của nó.
- Những chốt đầu vào và đầu ra trên các hành động được ra lệnh bắt đầu tại góc dưới bên phải của hành động chứa, và chúng được tiến hành theo chiều kim đồng hồ xung quanh hành động để hạ thấp phía bên phải. Chốt ra lệch này tương ứng với thứ tự của các tham số của hoạt động được gọi, với chốt đầu vào mục tiêu là chốt đầu tiên và (nếu bất cứ) kiểu hoạt động nào là chốt đầu ra cuối cùng tương ứng với kết quả trả về. Chốt đầu vào mục tiêu phản ánh đối tượng mục tiêu đối với yêu cầu hành động được gửi (ví dụ, lớp sở hữu hoạt động).
- Những kiểu chốt đầu vào và đầu ra không được thiết lập một cách tổng quát, bởi vì điều này có thể được sinh ra từ tham số tương ứng.
- Hoạt động này không sử dụng các luồng đối tượng để đơn giản hóa việc tạo ra sơ đồ. Thay vào đó, tên của các chốt đầu vào đầu ra trên các hành động được gán giá trị bằng một tham số, biến hay tính năng cấu trúc trong lớp sở hữu hoạt động nội dung có trong phạm vi. UML 2 cũng hỗ trợ điều này, và nó được sử dụng làm một sự kiện tham chiếu.
- Các nút tham số hoạt động (trên lề phải và trái của hoạt động) không được sử dụng. Thay vào đó, những tham số của hoạt động (nó phải tương ứng với các tham số của hoạt động cụ thể của nó) được tham chiếu trực tiếp lên chốt đầu vào đầu ra khi cần thiết. Những nút tham số hoạt động này

sẽ được sử dụng nếu các luồng đối tượng được sử dụng.

- Các phân vùng hoạt động được thiết lập để phản ánh các phần hay các cổng dịch vụ của các thành phần chứa hoạt động. Tất cả yêu cầu được tạo ra và tất cả các sự kiện được chấp nhận qua những phần này. Các phân vùng không được nêu ra trong trường hợp này, bởi vì đặc tính được phản ánh sẽ cung cấp thông tin để nhận dạng phân vùng.
- Các chốt đầu vào mục tiêu của các hành động được gọi không cần thiết phải được thiết lập. Theo luân phiên, phân vùng hoạt động phản ánh công dịch vụ nơi các lệnh gọi trong phân vùng đó được yêu cầu. Điều này được gọi là những phân vùng <<instance>> trong UML 2 và có ngữ nghĩa học được định nghĩa rõ ràng. Các chốt đầu vào mục tiêu cũng có thể được thiết lập, nhưng điều này không bắt buộc.
- Chốt `returnInformation` của hành động Accept Call được xử lý giống như chốt đầu vào mục tiêu của hành động được gọi. Nó cũng là cổng được phản ánh bởi phân vùng hoạt động, và phản ánh điểm tương tác mà thông qua đó lệnh gọi sẽ được chấp nhận.
- Các biểu thức chỉ định được đưa ra cùng với các hành động mờ, nơi tên của hành động chứa một biểu thức chỉ định tham chiếu tới các biến, tham số, và các tính năng cấu trúc trong phạm vi. Giá trị `lvalue` và `rvalue` trong câu lệnh chỉ định được phân cách bởi dấu **:=** (dấu hai chấm và dấu bằng).
- Các biểu thức bảo vệ trên các luồng đối tượng và điều khiển là các biểu thức Java hay XPath Boolean tham chiếu tới các biến, tham số, và tính năng cấu trúc trong phạm vi hoạt động.
    - *Dữ liệu* trên một luồng đối tượng được tham chiếu bằng tên của luồng đó.
    - *Kiểu* của dữ liệu trên một luồng đối tượng được quyết định bởi nguồn của nó.

Những nguyên tắc này được sử dụng để đơn giản hóa việc mô hình hóa hoạt động, đơn giản hóa các sơ đồ hoạt động, và để tương ứng tốt hơn với BPEL sẽ được sinh ra từ chúng.

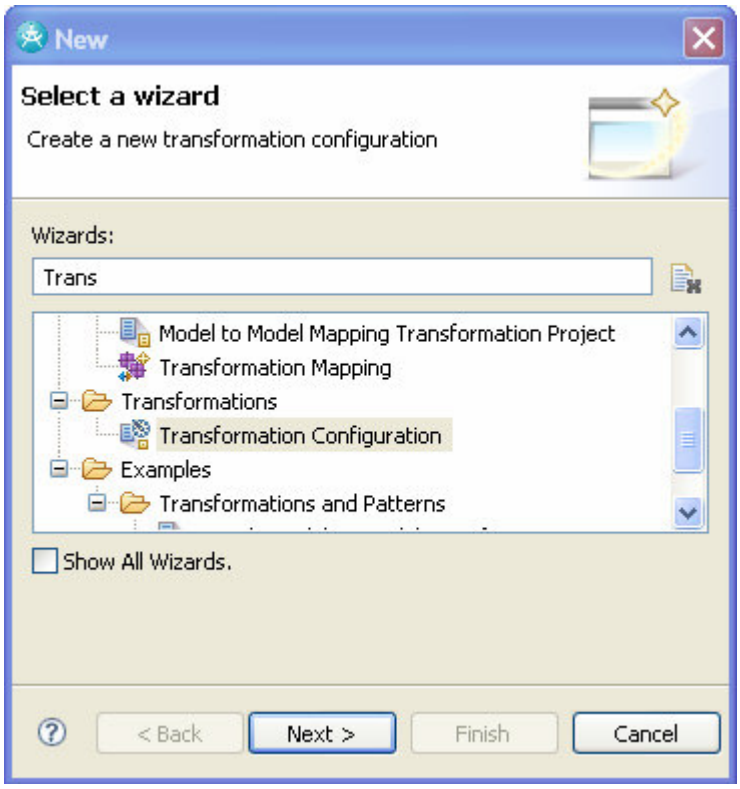
# Dịch sang các dịch vụ Web

Sự chuyển đổi yêu cầu sử dụng một cấu hình chuyển đổi.

### Cấu hình công cụ chuyển đổi

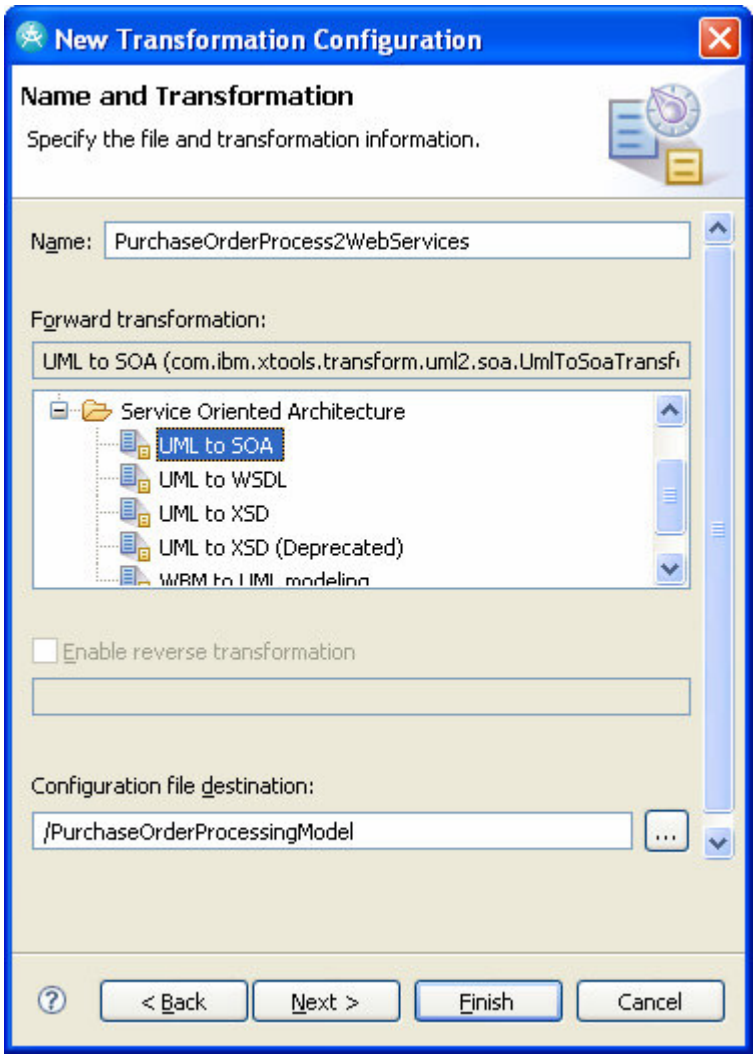
Bạn có thể tạo ra một chuyển đổi bằng cách chọn **File > New > Other > Transform Configuration** (Hình 12).

**Hình 12. Tạo mới một cấu hình chuyển đổi**




Chúng ta sẽ sử dụng một công cụ chuyển đổi UML-to-SOA cho ví dụ này, như Hình 13 cho thấy.

**Hình 13. Chọn công cụ chuyển đổi UML-to-SOA**



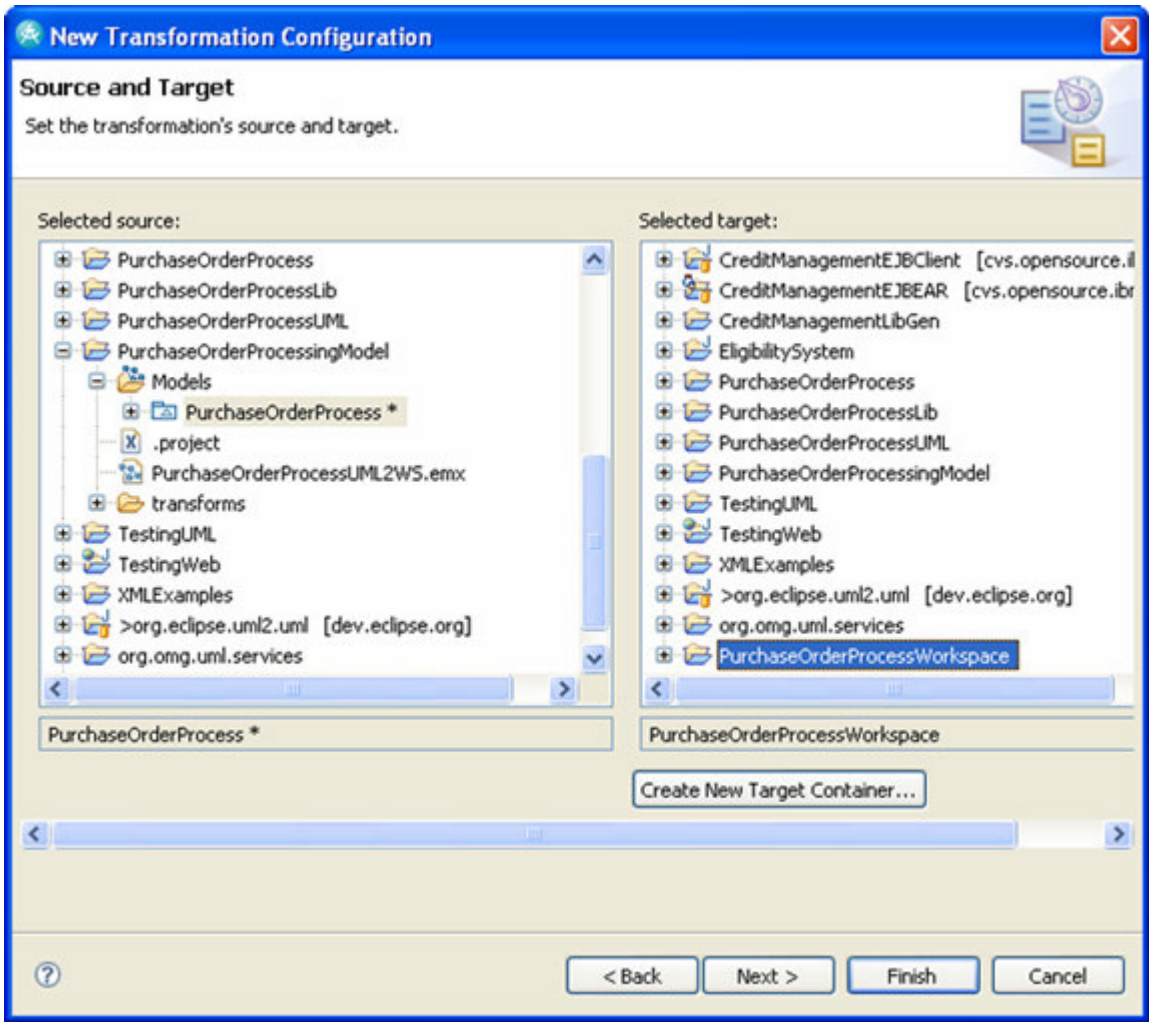

Cấu hình của phần lớn các công cụ chuyển đổi gồm ba phần cơ bản:

1. Chọn các yếu tố nguồn chuyển đổi
2. Chọn (hay tạo ra sau đó chọn) những yếu tố mục tiêu
3. Cấu hình những đặc tính chuyển đổi

Những yếu tố nguồn được chấp nhận được định nghĩa bằng công cụ chuyển đổi cụ thể được lựa chọn. Nhìn chung, cách tốt nhất là biến đổi toàn bộ các mô hình, không nên biến đổi từng phần của các mô hình. Điều này đảm bảo rằng những mô hình, phản ánh những tài nguyên được nhân bản riêng lẻ, được đối xử như là *các đơn vị biên dịch* đối với các chuyển đổi mô hình. Điều này giúp đơn giản hóa quản lí mô hình và việc xử lý sự phụ thuộc chuyển đổi bằng việc đảm bảo rằng sự thay đổi trong các tài nguyên trong vùng làm việc tương ứng với trình tạo và các đenta chuyển đổi nên được xử lý để đảm bảo rằng những công cụ được sinh ra đồng bộ với các yếu tố nguồn của chúng. Ví dụ, bạn sẽ không nghĩ tới việc chọn một phương thức riêng lẻ trong một lớp Java và chỉ biên dịch phương thức đó, và sau đó chèn nó vào những mã trình byte cho phần còn lại của lớp. Điều này sẽ làm không thể quản lí được trong một môi trường thay đổi rất nhanh, và nó còn yêu cầu bạn thay vì yêu cầu trình tạo và trình biên dịch, để biết tất cả các phụ thuộc mà có thể cần được biên như một kết quả của sự thay đổi.

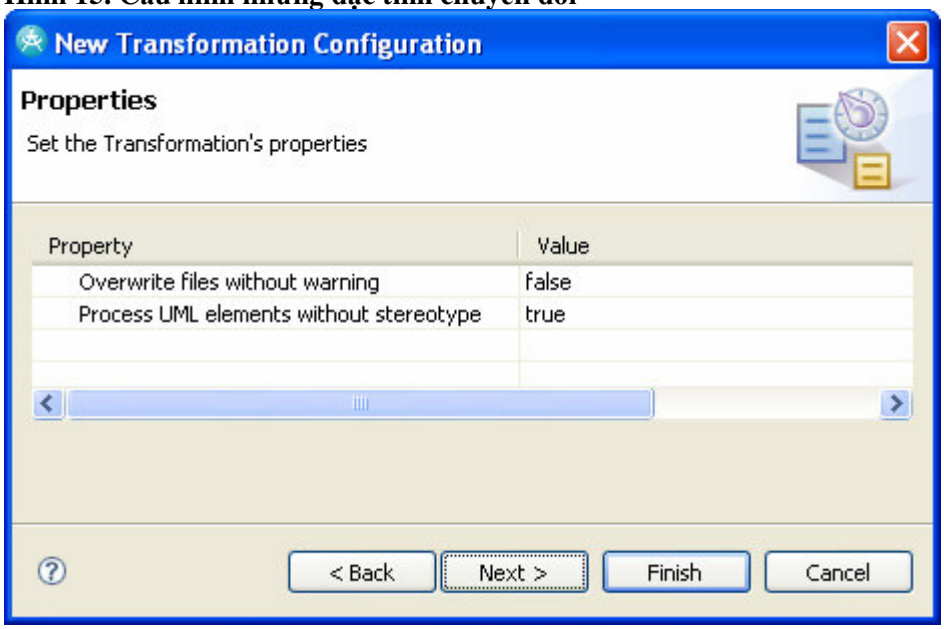
Trong ví dụ này, mô hình PurchaseOrderProcess là nguồn, và chúng ta đã tạo mới một dự án Eclipse mục tiêu đơn giản. Tất cả các yếu tố mô hình được dịch được đưa vào dự án này (Hình 14).

**Hình 14. Cấu hình các mục tiêu và nguồn chuyển đổi**




Các tham số cấu hình chuyển đổi phản ánh những tùy chọn không thích hợp đối với các đánh dấu trong mô hình. Điển hình, điều này điều khiển các sự tùy chọn toàn diện hơn là những sự tùy chọn áp dụng cho yếu tố mô hình cụ thể. Công cụ chuyển đổi UML-to-SOA chỉ có ít sự tùy chọn chuyển đổi, như Hình 15 cho thấy.

**Hình 15. Cấu hình những đặc tính chuyển đổi**



Xử lý các yếu tố UML với những lớp mẫu không được thiếp lập là *True* có nghĩa rằng hồ sơ IBM Software Services là tùy chọn hiện thời. Các kiểu dữ liệu, thành phần, và các hoạt động được dịch sang các giải pháp dịch vụ Web mà không yêu cầu các lớp mẫu `<<message>>`, `<<service>>`, hay `<<serviceProvider>>`, hoặc các lớp mẫu có thể được lược bỏ các yếu tố mô hình cụ thể không cần đến những đặc tính có thêm của chúng.

Điều này hoàn thiện cấu hình chuyển đổi của mô hình PurchaseOrderProcess thành một giải pháp những dịch vụ Web sẽ được đặt trong dự án PurchaseOrderProcessWorkspace.

## Chạy công cụ chuyển đổi

Thi hành trình chuyển đổi `PurchaseOrderProcess2WebServices` hoàn toàn đơn giản:

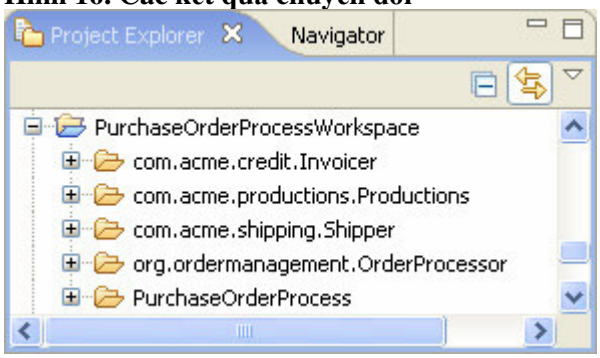
1. Chọn tệp cấu hình chuyển đổi **PurchaseOrderProcess2WebServices.tc** đã được tạo ra trong phần trước.
2. Trên thực đơn hiện ra, chọn **Transform > UML-to-SOA**.

Công cụ chuyển đổi được lựa chọn trong cấu hình chuyển đổi được thi hành, bằng phương thức đó chuyển đổi mô hình nguồn thành công cụ dịch vụ Web và đặt kết quả vào dự án.

**Mẹo:**
Nếu mô hình thay đổi, bạn sẽ cần chạy lại công cụ chuyển đổi này.

Những kết quả của sự chuyển đổi được biểu thị trong Hình 16 sử dụng Modeling perspective trong Project Explorer.

**Hình 16. Các kết quả chuyển đổi**




## Kiểm tra kết quả

Cấu hình chuyển đổi UML-to-SOA yêu cầu một dự án mục tiêu trong đó nó chứa tất cả những công cụ được phát sinh. Dự án này là một dự án Eclipse đơn giản chứa cả một thư viện WebSphere Integration Developer lẫn các dự án mô đun, như được miêu tả trong những phần phụ dưới đây.

- Các dự án *thư viện* chứa các đối tượng nghiệp vụ, các giao diện và các mô đun xuất khẩu mà được chia sẻ bởi các dự án khác.
- Các dự án *mô đun* chứa một cài đặt mô đun cho mỗi thành phần cung cấp dịch vụ trong mô hình các dịch vụ UML.

Bạn có thể nhập khẩu các dự án này vào vùng làm việc WebSphere Integration Developer của bạn. Tùy theo, bạn có thể sử dụng dự án mục tiêu như một vùng làm việc chứa tất cả các dự án được sinh ra cho một bộ những chuyển đổi UML-to-SOA của các mô hình SOA liên quan.

1. Đơn giản hãy khởi động WebSphere Integration Developer và chọn thư mục mục tiêu như vùng làm việc của bạn.
2. Sau đó, nhập khẩu tất cả các dự án trong thư mục đó vào trong vùng làm việc đó.

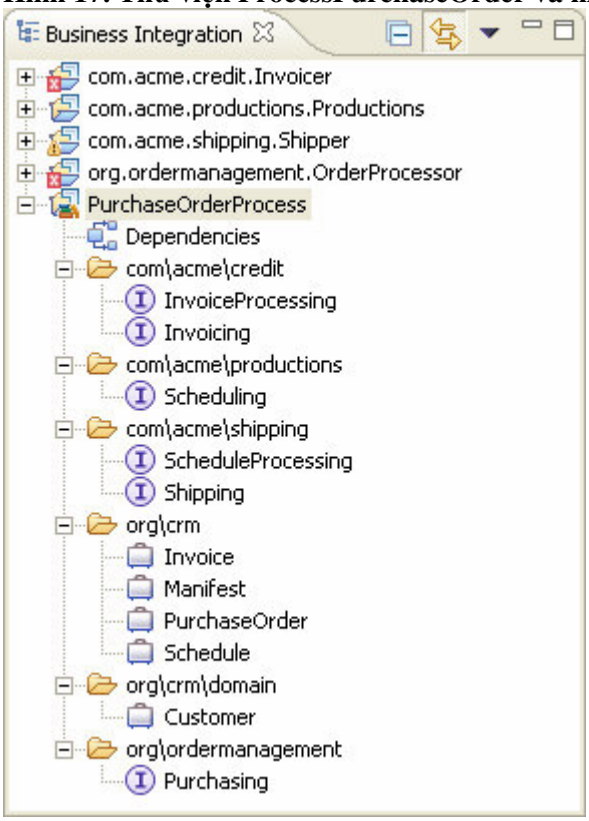
**Mẹo:**
Bạn có thể tắt chức năng Automatic Build cho đến sau khi tất cả các dự án được nhập khẩu xong, để đẩy nhanh quá trình nhập khẩu.

# Mô hình và các thư viện

Mỗi mô hình trong nguồn được lựa chọn của cấu hình chuyển đổi được dịch thành một thư viện WebSphere Integration Developer có cùng tên với mô hình. Thư viện này chứa một yếu tố XSD cho mỗi lớp và kiểu dữ liệu trong các mô hình nguồn, và một định nghĩa WSDL cho mỗi giao diện UML. Các thư viện này xác định các đối tượng nghiệp vụ và các giao diện được sử dụng bởi tất cả các mô đun WebSphere được sinh ra từ những thành phần trong nguồn được lựa chọn của cấu hình chuyển đổi.

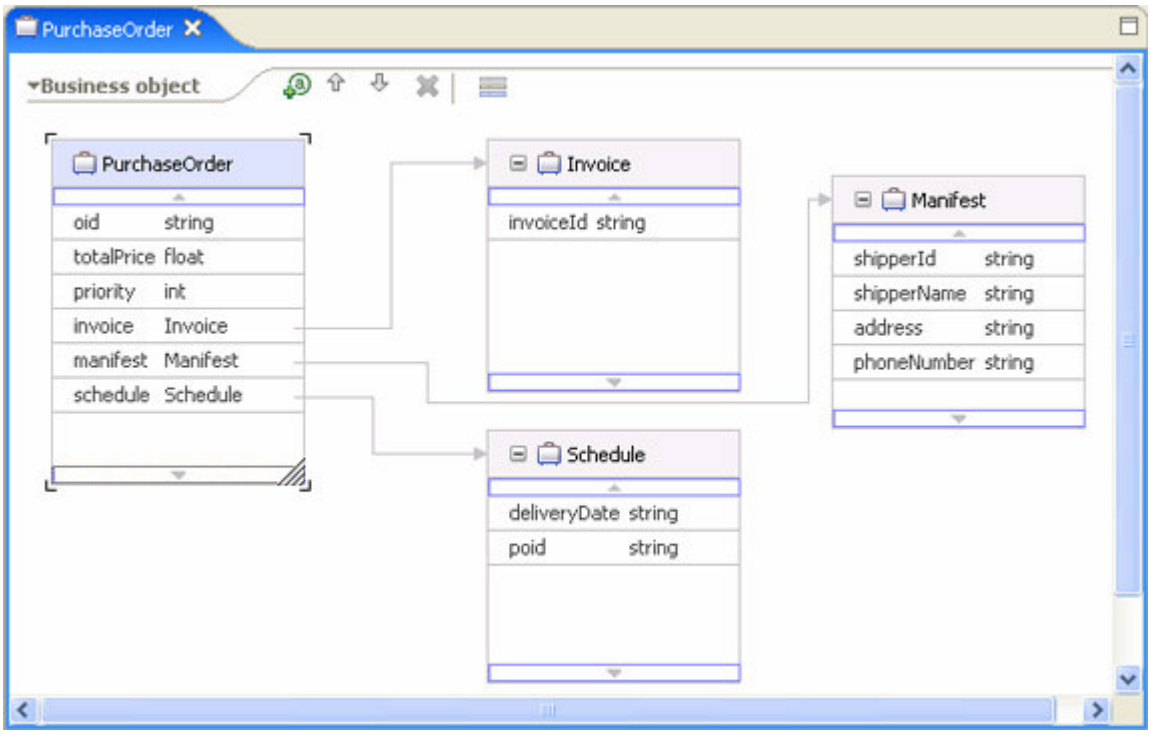
Hình 17 biểu thị thư viện và các dự án phát sinh được nhập khẩu trong WebSphere Integration Developer với PurchaseOrderProcess được mở rộng để biểu thị các giao diện và các đối tượng được sinh ra. Chú ý rằng những thư mục và tên vùng trong WebSphere phải tương ứng với gói cấu trúc trong mô hình các dịch vụ UML. Điều này đảm bảo quản lí tên vùng một các nhất quán và hỗ trợ dùng lại trong các tài nguyên và công cụ.

**Hình 17. Thư viện ProcessPurchaseOrder và những đối tượng và giao diện của thư viện này**



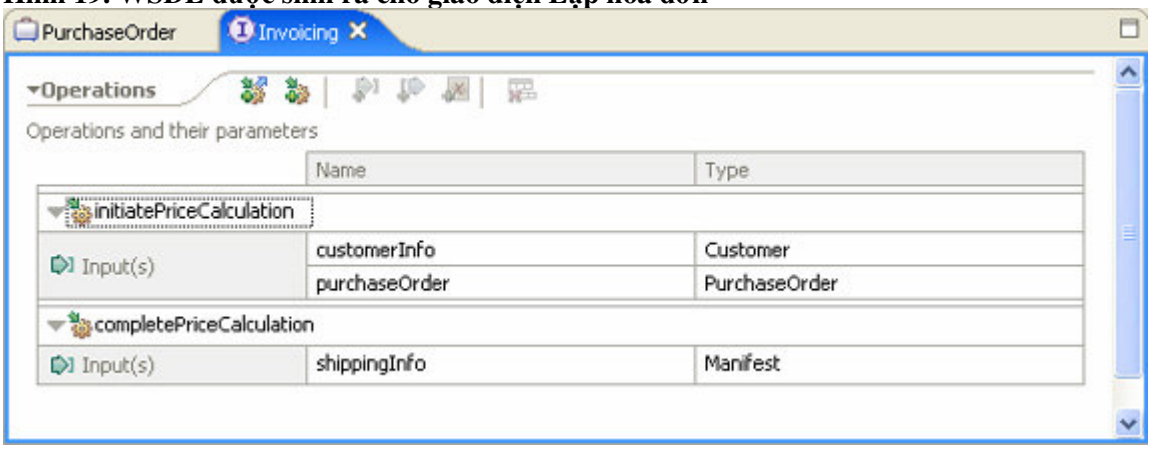

Chúng ta hãy tìm hiểu sâu hơn về các đối tượng và các giao diện nghiệp vụ và so sánh chúng với các yếu tố nguồn UML của chúng. Hình 18 sử dụng trình soạn thảo WebSphere Integration Developer Business Object được mở trên đối tượng nghiệp vụ PurchaseOrder để biểu thị các XSD được sinh ra từ mô hình dữ liệu dịch vụ được biểu thị trong Hình 2. Như bạn có thể thấy, các XSD tương ứng với các kiểu dữ liệu `<<message>>` nguồn của chúng. Nhấn chuột lên hình ảnh để xem nguồn được tạo ra.

**Hình 18. Các XSD được sinh ra từ mô hình dữ liệu dịch vụ**




Mỗi giao diện UML được chuyển đổi thành một WSDL portType. WSDL sinh ra cho giao diện Lập hóa đơn trong Hình 7 được biểu thị trong Hình 19. Nhấn chuột lên hình để xem nguồn WSDL được sinh ra. Xem lại, WSDL trông giống với giao diện UML.

**Hình 19. WSDL được sinh ra cho giao diện Lập hóa đơn**




# Các thành phần và các hệ thống mô đun

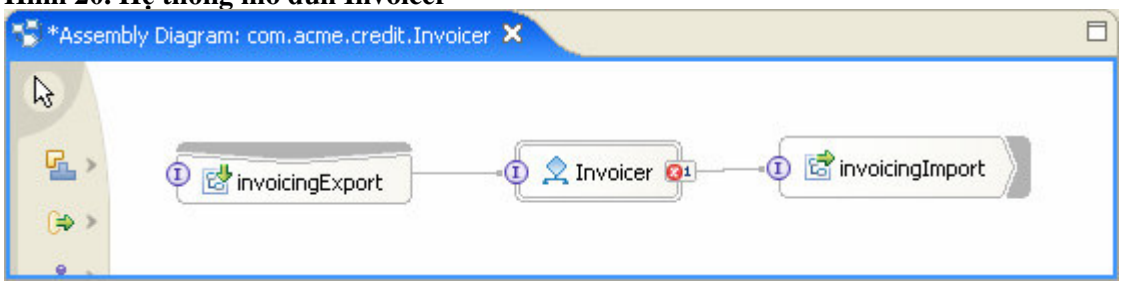
Mỗi thành phần cung cấp dịch vụ trong mô hình các dịch vụ UML được chuyển đổi thành một mô đun WebSphere Integration Developer. Vẫn chưa có một chuẩn các dịch vụ Web cho những người sử dụng dịch vụ linh kiện (khách hàng) và các nhà cung cấp. Do vậy, trình soạn thảo WebSphere Integration Developer Version 6.0.2 sử dụng một quyền sở hữu riêng, một phiên bản trước đây của Service Component Architecture (SCA). Các hệ thống mô đun được ghi lại trong các tệp **.component** sử dụng Service Component Description Language (SCDL), một ngôn ngữ tài liệu XML dành cho SCA. Một số công ty đang hợp tác với nhau để phát triển một chuẩn cho SCA. Xem trang Web Open SOA để biết thêm chi tiết.

Công cụ chuyển đổi UML-to-SOA tạo ra các mô đun WebSphere Integration Developer cho mỗi thành phần cung cấp dịch vụ để tối đa hóa tái sử dụng. Các mô đun SCA không thể được đấu nối từ các mô đun khác. Tuy nhiên, chúng có thể nhập khẩu các dịch vụ từ các mô đun khác và sử dụng chúng một các gián tiếp. Do vậy, kết nối giữa các khách hàng dịch vụ và các nhà cung cấp trong UML được cài đặt như kết

nối giữa các mô đun xuất khẩu và nhập khẩu trong WebSphere Integration Developer. Ví dụ, giả sử nhà cung cấp dịch vụ Invoicer được biểu thị trong Hình 8.

Hình 20 cho thấy linh kiện mô đun WebSphere Integration Developer tương ứng. Các mô đun xuất khẩu và nhập khẩu được tạo ra cho mỗi cổng dịch vụ. SCA không thể hỗ trợ cả hai giao diện được cung cấp và yêu cầu cho cùng một điểm tương tác, vì vậy các yếu tố xuất khẩu và nhập khẩu được tạo ra cho các cổng dịch vụ cung cấp và yêu cầu các giao diện. Dịch vụ invoicingExport xuất khẩu ra giao diện Lập hóa đơn được cung cấp; ngược lại, dịch vụ invoicingImport nhập khẩu giao diện InvoiceProcessing được yêu cầu của cổng dịch vụ lập hóa đơn của nhà cung cấp dịch vụ Invoicer.

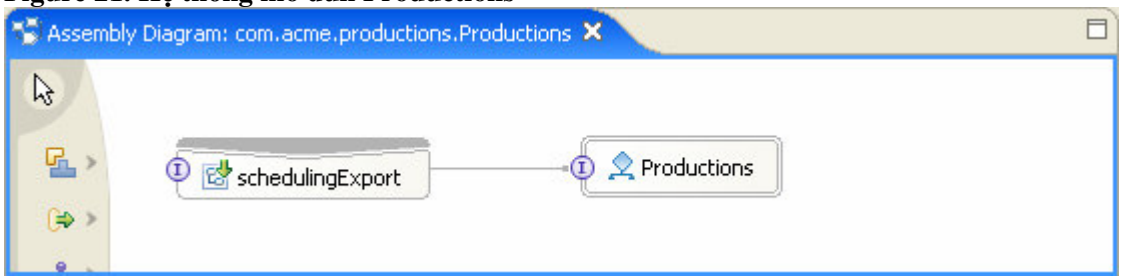
**Hình 20. Hệ thống mô đun Invoicer**




Chú ý tên mô đun. Một mô đun là một dự án Eclipse, nhưng bởi vì một mô đun là một yếu tố tái sử dụng, nó phải quản lí những xung đột về tên với những yếu tố tái sử dụng khác. Nguyên tắc được sử dụng bởi công cụ chuyển đổi UML-to-SOA là để tạo ra các tên dự án mô đun được dựa trên tên đã được xác định của thành phần cung cấp dịch vụ, khi được quyết định bởi gói chứa của nó. Điều này sẽ tạo ra các tên mô đun có chuỗi dài có thể gây ra một số vấn đề thời gian chạy trên nền tảng Windows do những giới hạn về độ dài của các URL. Những tên mô đun này có thể dễ dàng được chuyển thành những tên có độ dài ngắn hơn và thỏa mãn với những tên được quy định cho những nội dung tái sử dụng.

Thành phần Productions trong một hệ thống mô đun khác được biểu thị trong Hình 21. Mô đun này không có chức năng nhập khẩu, bởi vì cổng dịch vụ không yêu cầu bất cứ giao diện nào.

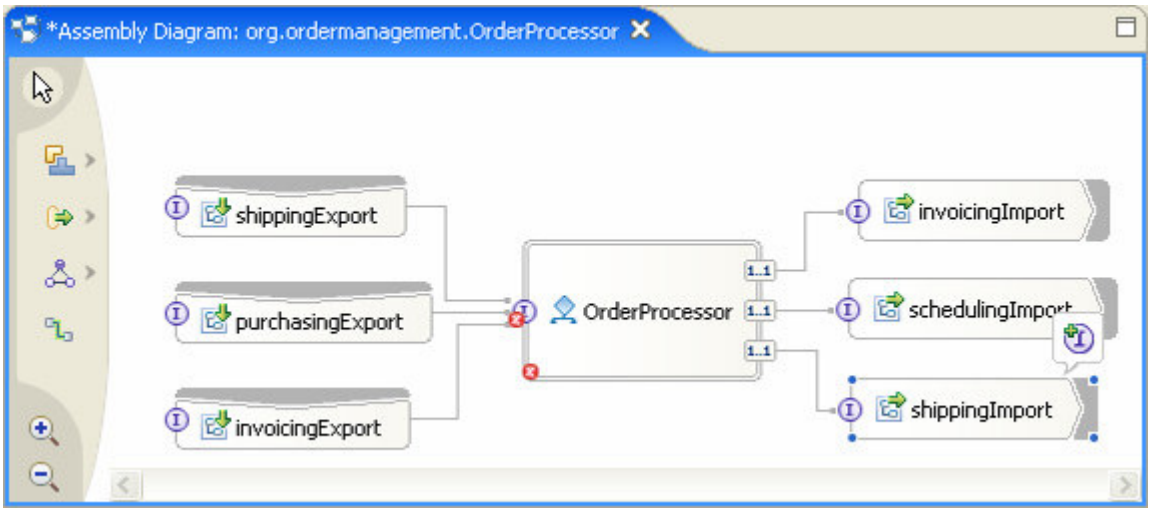
**Figure 21. Hệ thống mô đun Productions**




Cả hai mô đun này đều sử dụng một quá trình BPEL để cài đặt hiện thời các dịch vụ được cung cấp bởi các thành phần cung cấp dịch vụ tương ứng của chúng. Chi tiết về cách hoạt động của chúng sẽ được đề cập trong phần tới.

Hãy nhìn vào Hình 22 để xem hệ thống mô đun được tạo ra cho thành phần OrderProcessor trước đây, trong Hình 10.

**Hình 22. Hệ thống mô đun OrderProcessor**




Nhà cung cấp dịch vụ OrderProcessor cung cấp dịch vụ tiêu thụ, và có các tiêu chuẩn về yêu cầu đối với các dịch vụ lập hóa đơn, lập danh mục, và vận chuyển hàng. Các kênh dịch vụ kết nối với các thành phần khác hàng và nhà cung cung cấp trong Hình 10 được cài đặt khi các nhập khẩu trong mô đun OrderProcessor giáp với những xuất khẩu của các nhà cung cấp dịch vụ tương ứng. Điều này cho phép tái sử dụng mô đun hiệu quả trong WebSphere Integration Developer và giữ cho các mô hình dịch vụ độc lập với sự thay đổi của SCA. Khi SCA được tiêu chuẩn hóa, các mô hình dịch vụ UML sẽ không phải thay đổi; chỉ có công cụ chuyển đổi sẽ cần phải được cập nhật.

# Các hoạt động và các quá trình BPEL

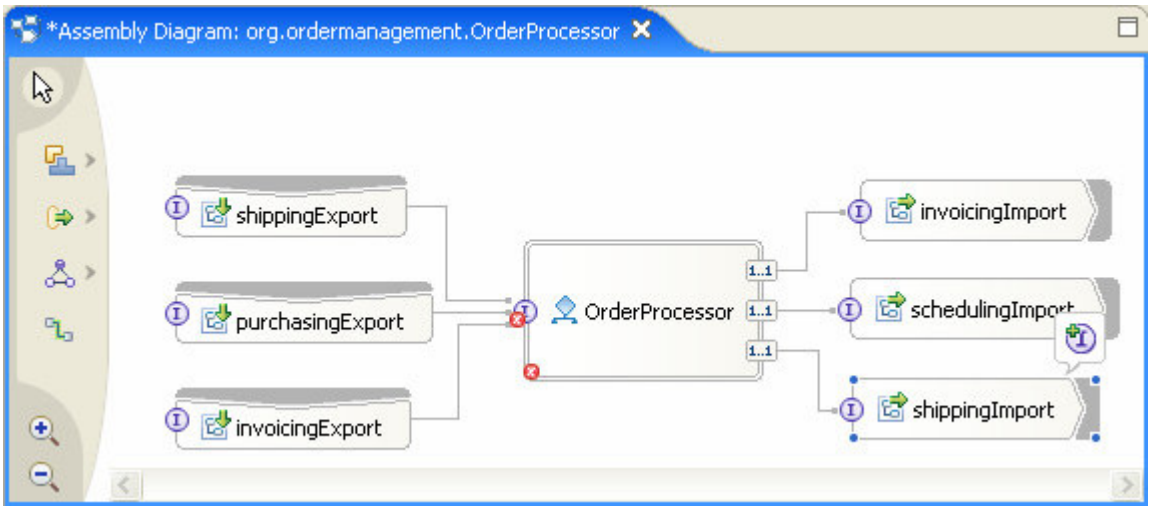
Mỗi hoạt động chức năng được cung cấp bởi một nhà cung cấp dịch vụ phải được cài đặt. Những cài đặt vừa được thiết kế trong UML bằng các sử dụng phương pháp ứng xử cho mỗi hoạt động vừa được thiết kế trong các hành động Accept Call trong một số ứng xử khác. Phương pháp sau đó cung cấp sự tách riêng có thêm giữa người sử dụng và nhà cung cấp bằng cách cho phép các nhà cung cấp quyết định thời điểm nó có thể sẵn sàng đáp ứng được yêu cầu, hơn là phải đáp ứng ngay lập tức khi hoạt động được yêu cầu bởi người sử dụng dịch vụ.

WebSphere Integration Developer SCA có phương thức tiếp cận khác. Mỗi xuất khẩu phải được kết nối với một thành phần trong hệ thống cung cấp một cài đặt cho các hoạt động trong giao diện đó. Các thành phần trong mỗi SCA có các kiểu cài đặt:

- Human Task
- Java
- Process
- Rules Group
- State Machine.

Nhà cung cấp dịch vụ OrderProcessor trong Hình 10 và Hình 22 cung cấp một hoạt động chức năng đơn lẻ thông qua dịch dụ tiêu thụ của nó để xử lý một thứ tự tiêu thụ. Sự cài đặt của hành động này là một hoạt động trong mô hình dịch vụ UML. Công cụ chuyển đổi UML-to-SOA tạo ra một quy trình BPEL từ hoạt động đó và sử dụng nó như là một cài đặt của hoạt động được xuất khẩu.

**Hình 23. Thực thi thành phần quy trình OrderProcessor**




Chú ý rằng quy trình BPEL trong Hình 23 rất giống với hoạt động được biểu thị trong Hình 11. Hướng dẫn chi tiết về cách hoạt động UML được chuyển đổi thành một quá trình BPEL có thể được tìm thấy trong phần Trợ giúp của công cụ Rational Software Architect.

### Tách giao diện với cài đặt

Như đã miêu tả ở trên, những thi hành WebSphere Integration Developer SCA đã cung cấp các khả năng bằng cách kết nối với một xuất khẩu cung cấp những giao diện cho một thành phần cài đặt các hoạt động được cung cấp. Bởi vì một thành phần có một kiểu cài đặt, tất cả các hoạt động được cung cấp bởi tất cả giao diện của xuất khẩu đó phải được cài đặt theo cùng một cách. Điều này kết nối kiểu cài đặt với tất cả các giao diện của một xuất khẩu. (Một xuất khẩu có thể được kết nối và cài đặt bởi chỉ một thành phần.) Nếu người phát triển muốn thay đổi kiểu cài đặt của một hoạt động nhất định (ví dụ, từ một nhiệm vụ nhân tạo đến một dịch vụ tự động được thi hành trong Java), các giao diện phải được kết nối để cho phép các xuất khẩu khác nhau có thể được kết nối với các kiểu cài đặt thành phần khác nhau. Lần lượt, điều này yêu cầu thay đổi đối với tất cả các người sử dụng các giao diện. Sự kết nối này của giao diện thiết kế cho kiểu cài đặt có thể hạn chế sự linh hoạt nghiệp vụ mà các giải pháp SOA hỗ trợ.

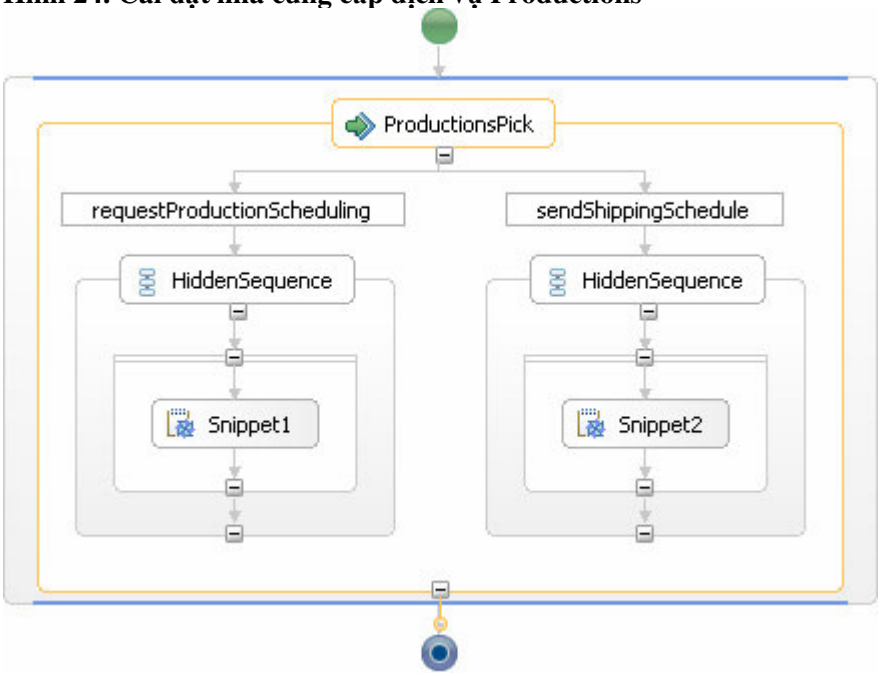
UML không có sự kết nối cài đặt và đặc điểm này. Mỗi hoạt động được cung cấp có thể có một phương pháp ứng xử là một hoạt động, tương tác, máy trạng thái, hay (mã trình) ứng xử mờ. Người lập mô hình sẽ thiết kế sự cài đặt của mỗi hoạt động theo một cách độc lập. Điều này cho ra kết quả trong những tình huống (được biểu thị trong Hình 4) nơi mà cùng một thành phần cung cấp dịch vụ sử dụng các kiểu ứng xử khác nhau cho các hoạt động khác nhau được cung cấp thông qua cùng một giao diện. Chúng ta cần một số cách để dịch các nhà cung cấp dịch vụ này sang các dịch vụ Web.

Có một yếu tố khác cũng cần xem xét. Trong UML, các thành phần được minh họa bằng các ví dụ, hơn là các ứng xử mà chúng sở hữu. Do vậy, việc xác minh trường hợp và chu kỳ là giống nhau đối với tất cả các ứng xử trong cùng thành phần. Thêm nữa, thành phần thiết lập một ngữ cảnh hay phạm vi cho tất các ứng xử mà nó sở hữu, bằng cách đó cho phép các ứng xử đó chia sẻ truy cập với các thành phần trạng thái (các đặc tính và các cổng). Vì thế, khi dịch thành các dịch vụ Web, chúng ta phải có công cụ nào đó để quản lí tính đồng nhất, chu kỳ, và trạng thái được chia sẻ này để có thể cài đặt các ngữ nghĩa học UML. Đây là nơi các quá trình Business Process Execution Language (BPEL) đi vào (xem Tài nguyên để biết thêm thông tin chi tiết về BPEL).

Thay vì tạo ra một thành phần SCA riêng biệt thi hành mỗi ứng xử của nhà cung cấp dịch vụ trong hệ thống mô đun, chúng ta sẽ tạo ra một quá trình BPEL độc lập tương ứng với thành phần của nó. Bạn phải chú ý rằng tên của quá trình BPEL trong Hình 23 là **OrderProcessor,** giống như nhà cung cấp dịch vụ OrderProcessor, không giống với **processPurchaseOrder**, tên của hoạt động được cung cấp.

Chúng ta hãy tìm hiểu sâu hơn về Hình 4 để xem thành phần Productions được dịch thành các dịch vụ WebSphere Integration Developer Web. Chú ý rằng thiết kế cài đặt cho requestProductionScheduling sử dụng một Activity (với thông tin chi tiết không được hiện thị), nhưng sendShippingSchedule sử dụng một OpaqueBehavior với sự cài đặt được cung cấp trong mã trình Java. Hệ thống mô đun cho nhà cung cấp dịch vụ này được biểu thị trong Hình 21, và thành phần Productions Process được biểu thị trong Hình 24.

**Hình 24. Cài đặt nhà cung cấp dịch vụ Productions**




Một quá trình BPEL được tạo ra cho thành phần nhà cung cấp dịch vụ Productions. Những đặc tính nhận dạng của nhà cung cấp dịch vụ được sử dụng để định nghĩa sự tương quan cho tất cả hoạt động Invoke, Receive, và Reply trong quá trình này. Mỗi hoạt động được cung cấp bởi thành phần được cài đặt bởi một đoạn của quá trình này. Quá trình bắt đầu với một yếu tố chọn lọc được sử dụng để gửi đi mỗi yêu cầu hoạt động. Sau đó, một phạm vi được tạo ra cho mỗi hoạt động để cung cấp một vị trí cho các biến được định nghĩa trong trình ứng xử UML. Tiếp theo, phạm vi sẽ chứa kết quả của trình ứng xử được dịch. nếu trình ứng xử là một UML Activity, thì phạm vi sẽ chứa BPEL được sinh ra từ hoạt động đó. Nếu trình ứng xử là một OpaqueBehavior với ngôn ngữ Java, thì phần chính của trình ứng xử được sao chép vào một hoạt động được chia bởi Java trong phạm vi. Nếu trình ứng xử là một OpaqueBehavior với ngôn ngữ HTML hoặc ngôn ngữ Java™Server Pages (JSP), thì hoạt động Human Task được thêm vào trong phạm vi.

Điều này cung cấp sự cách ly hoàn toàn của các giao diện và các cài đặt của chúng. Ví dụ, nếu người lập mô hình hay người phát triển quyết định thay đổi một cài đặt hoạt động dịch vụ từ nhiệm vụ nhân tạo thành một dịch vụ java tự động, chỉ yếu tố nhiệm vụ nhân tạo trong phạm vi dành cho hoạt động đó là cần thay đổi. Các máy khách sẽ không biết rằng sự thực đã được thay đổi cho đến khi họ nhận thấy rằng dịch vụ chạy nhanh hơn và họ không phải thực hiện nhiều thao tác.

Cũng dễ hiểu rằng điều này đôi khi phức tạp và khó hiểu, và gần như phải thay đổi khi SCA được tiêu chuẩn hóa. Nhưng sự cách ly, chia rẽ của giải pháp liên quan, tái sử dụng và linh hoạt là cơ sở cho SOA, và hiểu được nó không phải lúc nào cũng dễ dàng.

## Hoàn thành giải pháp

Công cụ chuyển đổi UML-to-SOA không sinh ra những giải pháp hoàn chỉnh. Bởi vì việc đưa chi tiết cài đặt vào trong các mô hình khó hơn việc đưa nó vào các công cụ theo nền tảng bằng việc sử dụng các trình soạn thảo được tạo ra theo mục đích. Có một số tính năng của UML 2 Activities chưa được dịch sang BPEL. Những tính năng này sẽ được đưa thêm vào trong những phiên bản sau. Thông tin chi tiết về những tính năng được hỗ trợ trong những phiên bản cụ thể có trong phần Trợ giúp của Rational Software Architect.

Đặc biệt, WebSphere Integration Developer sẽ phải thực hiện một số hoặc tất cả các hoạt động phát triển như dưới đây.

1. **Thêm các mã trình Java cho những ứng xử mờ** nếu nó không được cung cấp trong mô hình. Ngay cả khi, các mã trình Java được cung cấp trong phần chính của ứng xử mờ, nó vẫn thường có các lỗi xảy ra, bởi vì Content Assist và trình biên soạn Java vẫn chưa kết nối với những khả năng mô hình hóa của Rational Software Architect.

2. **Thêm sự tương quan cho những quá trình nghiệp vụ.** Sự tương quan định rõ những thông tin cần thiết để nhận dạng các ví dụ của một thành phần, và nó không được hỗ trợ trong công cụ chuyển đổi UML-to-SOA. Tính năng này cũng được hỗ trợ trong phiên bản sau.
3. **Tạo ra một giao diện người dùng (UI) dành cho các nhiệm vụ nhân tạo.** Người lập mô hình UML có thể sử dụng ngôn ngữ JSP hoặc HTML trong phần chính của một ứng xử mờ. Nhưng, như mã nguồn Java, điều này có thể chưa hoàn thiện hoặc không chính xác. Người phát triển tích hợp sẽ muốn sử dụng trình soạn thảo Human Task, Page Designer, cổng, hay các công cụ khác để tạo ra các nhiệm vụ nhân tạo hoàn thiện với góc nhìn tương đối để hoàn thiện UI của ứng dụng.
4. **Cấu hình mô hình màn hình.** Hiện tại, công cụ chuyển đổi UML-to-SOA không tạo ra bất cứ thông tin màn hình nào để đánh giá nghiệp vụ Key Performance Indicators (KPIs) có thể được mô hình hóa như những ràng buộc trên các dịch vụ và các nhà cung cấp dịch vụ. Người phát triển tích hợp có thể sử dụng công cụ Monitor Model Editor trong WebSphere Integration Developer để cấu hình những gì dữ liệu cần lựa chọn cho những cập nhật giả lập và cho các hệ đo mà đơn vị đo nghiệp vụ.

# Tóm tắt

Bài viết này đã kết thúc loạt bài viết về mô hình hóa các dịch vụ trong Rational Software Architect sử dụng hồ sơ IBM Software Services (xem **Thông tin liên quan về chủ đề này,** ở góc trên bên trái). Bài viết đầu tiên, **Mô hình hóa SOA: Phần 1. Sự nhận biết dịch vụ**, đã hướng dẫn cách kết hợp các mô hình quy trình nghiệp vụ và các dịch vụ để đưa ra các yêu cầu của hợp đồng để một bộ dịch vụ có thể được kết nối và xây dựng để hoàn thành một số mục tiêu. Các hợp đồng dịch vụ này thường được sử dụng để định ra những việc phải hoàn thành để đạt được các mục tiêu nghiệp vụ. Sau đó, Các kết hợp dịch vụ có thể được sử dụng trong việc nhận biết các dịch vụ cung cấp giá trị nghiệp vụ thực sự.

**Mô hình hóa SOA: Phần 2. Đặc điểm dịch vụ** đã chỉ ra cách mô hình hóa các thông tin chi tiết của một đặc điểm dịch vụ từ hình phối cảnh của nhà cung cấp. Một đặc điểm dịch vụ định nghĩa các thông tin cần thiết cho những khách hàng tiềm năng để quyết định nếu một dịch vụ được áp cho vấn đề của họ, và nếu như vậy, thì sẽ sử dụng nó như thế nào. Một đặc điểm dịch vụ cũng định nghĩa nhà cung cấp của một dịch vụ cần làm gì để cài đặt dịch vụ.

Các bài viết tiếp theo, **Mô hình hóa SOA: Phần 3. Thực hiện dịch vụ** và **Mô hình hóa SOA: Phần 4. Cấu trúc dịch vụ** đã chỉ ra cách thiết kế và mô hình hóa các thành phần cung cấp hoặc yêu cầu các dịch vụ và miêu tả cách các hoạt động dịch vụ được thi hành. Hai bài viết này cũng miêu tả cách các nhà cung cấp dịch vụ đưa ra các hợp đồng mà họ thực hiện để kết nối với các nhà cung cấp dịch vụ nhằm đạt được những mục tiêu, quá trình, và yêu cầu về nghiệp vụ.

Bài viết cuối cùng miêu tả cách thi hành các dịch vụ trên nền tảng các dịch vụ Web được cung cấp bởi IBM WebSphere Integration Developer bằng cách sử dụng công cụ chuyển đổi IBM UML-to-SOA. WebSphere Integration Developer hỗ trợ một mô hình lập trình SOA phù hợp với những thiết kế thực hiện, đặc điểm, và nhận biết có trong Rational Software Architect. Bằng việc sử dụng WebSphere Integration Developer, bạn có thể hoàn thành lập trình các dịch vụ, và sau đó tạo ra một UI cho những nhiệm vụ nhân tạo, chạy các kiểm tra, và triển khai giải pháp SOA của bạn.

---

# Tải về

| Mô tả | Tên | Kích thước | Phương thức tải |
|---|---|---|---|
| RSM sample model | RSM_sample_model.zip | 60KB | **HTTP** |

→ Thông tin về phương thức tải

# Tài nguyên

**Học tập**

- Mô hình hóa SOA: Phần 1. Xác định dịch vụ của tác giả Jim Amsden là bài viết đầu tiên trong loạt 5 bài viết về phát triển phần mềm dựa trên nền tảng hướng dịch vụ (SOA). (IBM® developerWorks®, tháng Mười năm 2007).

- Mô hình hóa SOA: Phần 2. Đặc điểm dịch vụ của tác giả Jim Amsden là bài viết thứ hai trong loạt bài viết này. (IBM® developerWorks®, tháng Mười năm 2007).

- Mô hình hóa SOA: Phần 3. Thực hiện dịch vụ của tác giả Jim Amsden là bài viết thứ ba trong loạt bài viết này. (IBM® developerWorks®, tháng Mười năm 2007).

- Mô hình hóa SOA: Phần 4. Các thành phần tạo lên dịch vụ của tác giả Jim Amsden là bài viết thứ tư trong loạt bài viết này. (IBM® developerWorks®, tháng Mười năm 2007).

- Mô hình hóa và kiến trúc định hướng dịch vụ : Cách nhận dạng, chỉ định, và thực hiện các dịch vụ cho giải pháp SOA của bạn của tác giả Ali Arsanjani là bài viết về phương thức IBM Global Business Services' Service Oriented Modeling and Architecture (SOMA) (IBM® developerWorks®, tháng Mười Một năm 2004).

- Mô hình hóa dịch vụ IBM Business, một bài viết developerWorks của tác giả Jim Amsden (tháng Mười Hai năm 2005), miêu tả mối quan hệ giữa mô hình hóa quá trình nghiệp vụ và mô hình hóa dịch vụ để đạt được những lợi ích cho cả hai.

- "Sử dụng phát triển hướng mô hình và thiết kế dự trên kiểu mẫu để thiết kế SOA: Phần 2. Thiết kế dựa trên kiểu mẫu," Phần 2 của loạt hướng dẫn gồm 4 phần trên IBM developerWorks (2007).

- "Thiết kế các dịch vụ SOA với Rational Software Architect," loạt hướng dẫn gồm 4 phần trên IBM developerWorks (2006-2007).

- "Mô hình hóa kiến trúc định hướng dịch vụ bằng Rational Software Architect: Phần 3. Mô hình hóa hệ thống ngoại vi," Phần 3 của của loạt hướng dẫn gồm 6 phần trên IBM developerWorks (2007).

- Mô hình hóa các giải pháp định hướng dịch vụ là bài viết của tác giả Simon Johnston miêu tả phương pháp tiếp cận với mô hình hóa dịch vụ định hướng sự phát triển của UML Profile đối với Software Services và của RUP đối với trình cài thêm SOA (developerWorks, tháng Bảy năm 2005).

- UML 2.0 Profile for Software Services, cũng là bài viết của tác giả Simon Johnston (developerWorks, tháng Tư 2005), miêu tả hồ sơ UML cho dịch vụ phần mềm, một hồ sơ cho UML 2.0 cho phép đối với sự mô hình hóa của các dịch vụ, kiến trúc định hướng dịch vụ (SOA), và các giải pháp định hướng dịch vụ. Hồ sơ đã được thi hành trong IBM Rational Software Architect.

- Mô hình lập trình SOA để cài đặt các dịch vụ Web, Phần 1: Giới thiệu về mô hình lập trình IBM SOA, là bài viết của tác giả Donald Ferguson và Marcia Stockton (developerWorks, tháng Sáu năm 2005), miêu tả mô hình lập trình IBM dành cho Service-Oriented Architecture (SOA), nó giúp những người không phải lập trình viên có thể tạo ra và tái sử dụng các tài sản IT. Mô hình bao gồm các kiểu thành phần, kết nối, các khuôn mẫu, bộ điều hợp ứng dụng, đại diện dữ liệu đồng bộ, và một Enterprise Service Bus (ESB). Đây là bài viết đầu tiên trong loạt bài viết về mô hình lập trình IBM SOA và yêu cầu những gì để chọn, phát triển, triển khai, và đề nghị những yếu tố mô hình lập trình.

- Service Data Objects đơn giản hóa và thống nhất cách các ứng dụng truy cập và thao tác dữ liệu từ các nguồn dữ liệu hỗn tạp.

- Xem Business Process Execution Languageg for Web Services để biết thêm chi tiết về đặc điểm BPEL 1.1.

- Hãy đăng ký bản tin developerWorks Rational. Bạn sẽ được cập nhật hàng tuần về các tài nguyên công nghệ mới nhất và những ví dụ tiêu biểu về Rational Software Delivery Platform.

- Truy cập vào hiệu sách công nghệ để biết thông tin về những cuốn sách về các chủ để công nghệ khác.

**Lấy sản phẩm và công nghệ**

- Tải xuống trình cài thêm Rational Unified Process dành cho phương pháp SOMA: IBM RUP for Service-Oriented Modeling and Architecture. Bạn có IBM Rational Method Composer mới có thể cài đặt trình cài thêm này được.
- Tải xuống trình cài thêm RUP dành cho SOA, trình cài thêm Rational Unified Process dành cho việc mô hình hóa các dịch vụ sử dụng IBM Software Services Profile.
- IBM SOA Sandbox IBM SOA Sandbox cung cấp một pha trộn của các bản phần mềm dùng thử phiên bản đầy đủ và các môi trường được đặt "trực tuyến" ở đó bạn có thể xem các hướng dẫn sử dụng và hướng dẫn cấu trúc.
- Tải xuống phiên bản dùng thử của IBM Rational Software Architect V7.
- Tải xuống phiên bản dùng thử của các sản phẩm của IBM và làm quen với các công cụ phát triển ứng dụng và các sản phẩm trung gian từ DB2®, Lotus®, Rational®, Tivoli®, và WebSphere®.

**Thảo luận**

- Xem qua blog developerWork và tham gia vào cộng đồng developerWorks.
- và diễn đàn Rational Software Architect, Data Architect, Software Modeler, Application Developer and Web Developer: Đưa ra những câu hỏi về Rational Software Architect.

# Đôi nét về tác giả

Jim Amsden là một nhân viên kỹ thuật lâu năm tại IBM với hơn 20 năm kinh nghiệm thiết kế và phát triển các ứng dụng và công cụ cho công nghiệp phát triển phần mềm. Ông có bằng Thạc sĩ về Khoa học Máy tính tại trường Boston. Sở thích của ông là phát triển các yêu cầu, lập trình mạng, phát triển hướng dịch vụ, J2EE UML, và các kiến trúc định hướng dịch vụ. Ông là đồng tác giả của cuốn "Enterprise Java Programming with IBM WebSphere". Hiện tại, Jim tập trung tìm cách tích hợp các công cụ để hỗ trợ tốt hơn cho quá trình phát triển phối hợp.