HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM VĂN CƯỜNG, NGUYỄN TRỌNG KHÁNH

BÀI GIẢNG
PHÁT TRIỂN PHẦN MỀM HƯỚNG DỊCH VỤ

HÀ NỘI 2020

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ

B2B	Business-to-Business
B2C	Business-to-Consumer
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPML	Business Process Management Language
ebXML	Electronic Business eXtensible Markup Language
EDI	Electronic Data Interchange
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
OWL	Web Ontology Language
OWL-S	OWL service
RDF	Resource Description Framework
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery, and Integration protocol
URI	Uniform resource identifier
WSCI	Web Service Choreography Interface
WSDL	Web Services Description Language
WSML	Web Service Markup Language
WWW	World Wide Web
XACML	eXtensible Access Control Markup Language
XML	EXtensible Markup Language
XPDL	XML Process Definition Language

CHƯƠNG 1 GIỚI THIỆU

1.1 TIẾN HÓA CỦA WEB HIỆN NAY

Mặc dù hiện tại World-Wide Web (WWW) được sử dụng bởi con người. Tuy nhiên, theo hầu hết các chuyên gia cũng như nhà sáng lập ra WWW, Tim Berners-Lee, đã cho rằng WWW sẽ phải tiến hóa để các hệ thống máy tính có thể sử dụng được. Sự tiến hóa này được trông đợi từ việc thiết kế, phát triển và triển khai các dịch vụ Web. Cụm từ dịch vụ Web để chỉ các tiêu chuẩn cơ bản cho phép tích hợp các ứng dụng web mà các ứng dụng này có thể sử dụng các thành phần khác nhau để tạo thành một dịch vụ.

Các thành phần của Web ngày càng trở nên phổ biến (ubiquitous), phân tán (distributed), không đồng nhất (heterogeneous) và có xu thế tự trị (autonomous). Ban đầu, các thành phần của môi trường này là các trang Web. Nói cách khác, các trang Web cung cấp cả nội dung và dịch vụ. Nhưng ngày một nhiều trong tương lai Web cũng trở nên ngày càng tiến hóa và các thành phần của nó ngày càng có khả năng tùy biến. Trong đó các thành phần của Web cho phép người dùng dễ dàng tương tác và thậm chí các thành phần này có thể tương tác với nhau như những chương trình máy tính. Cụ thể, web đã tiến triển từ thế hệ đầu tiên với công nghệ trình duyệt thông thường, và cho đến nay là thế hệ thứ 4 với các dịch vụ web dựa trên ngữ nghĩa. Bảng 1.1 minh họa điều này.

Bảng 1.1: Tiến hóa web

Thế hệ web	**Phạm vi**	**Công nghệ**	**Ví dụ thực tế**
1	Tất cả	Trình duyệt	Các trang HTML
2	Chương trình	Giao tác với màn hình	Phát sinh nội dung HTML một cách có hệ thống
3	Theo tiêu chuẩn	Các dịch vụ web	Các dịch vụ được mô tả hình thức
4	Ngữ nghĩa	Các dịch vụ web ngữ nghĩa	Các dịch vụ được mô tả theo ngữ nghĩa

1.2 DỊCH VỤ WEB

Cũng giống như khái niệm *đối tượng* (object) của thế hệ trước đây, khái niệm *dịch vụ* (service) là từ được sử dụng rộng rãi hiện nay. Khái niệm dịch vụ có thể có các ý nghĩa khác nhau đối với những người khác nhau. Nhưng nhìn chung dịch vụ Web (Web service) có thể được định nghĩa là:

- Một phần của nhà cung cấp (piece of business) có thể truy cập thông qua Internet bằng cách sử dụng các tiêu chuẩn mở (định nghĩa của công ty Microsoft);
- Bao phủ, lỏng lẻo, giao tác bằng các hàm thông qua các giao thức chuẩn trên Web (định nghĩa của công ty DestiCorp);
- Thành phần phần mềm kết nối lỏng lẻo cùng tương tác với nhau thông qua các chuẩn công nghệ Internet (định nghĩa của công ty Gartner);

- Một ứng dụng phần mềm nhận dạng bằng một định dạng tài nguyên thống nhất (Uniform Resource Identifier), có giao diện và bắt buộc có khả năng được xác định, mô tả, và phát hiện bởi bằng ngôn ngữ đánh dấu mở rộng (XML), và hỗ trợ tương tác trực tiếp với các ứng dụng phần mềm khác sử dụng các thông điệp XML dựa trên các giao thức Internet (định nghĩa của tổ chức W3C).

1.3 WEB NGỮ NGHĨA

Tim Berners-Lee, cha đẻ của World Wide Web, đã mô tả rằng: các thành phần của web sẽ không tập trung mà có xu hướng phân tán khắp nơi, không đồng nhất và tự trị. Đó chính là web ngữ nghĩa (*Web Semantics*). Thông tin trên web được đánh dấu (markup) lên để trình diễn và được hiển thị bằng một trình duyệt. Con người có thể giải thích nội dung của thông tin vì họ đã có nền tảng tri thức mà họ chia sẻ với những người tạo ra các trang web đó. Trừ khi các chương trình được tạo ra để biểu diễn và khai thác tri thức như vậy. Việc xử lý vấn đề này thường bị giới hạn vì các mã chương trình được lập trình sẵn (hard-coded); điều này không phù hợp với một thiết lập động (dynamic) vì các chi tiết về thông tin trên web có thể dễ dàng thay đổi theo thời gian.

Ví dụ, chúng ta có thể viết một chương trình sao chép màn hình (screen-scraping) để trích xuất (extract) giá một cuốn sách từ một trang kết quả tìm kiếm trên trang thương mại điện tử amazon.com. Chương trình này sẽ dựa vào cú pháp của các trang web đã được mã hóa theo một ngữ pháp hình thức (formal gramma). Bằng trực giác, một chương trình có thể được hướng dẫn để đọc giá từ các trang kết quả phân tích một cách thích hợp. Tùy thuộc vào cấu trúc của trang tại các trang web nhất định mà những hướng dẫn này có thể phải thay đổi mà không được báo trước (*ad hoc*). Ví dụ, thông tin về giá của cuốn sách A ở dòng thứ 2, cột 3 bảng thứ 4 trong khung (frame) thứ 5 vào hôm nay; nhưng hôm sau do việc bổ sung thêm một vài loại hàng hóa khác có thể khiến thông tin về giá của cuốn sách A không còn ở vị trí cũ nữa. Mặc dù có một số công cụ hiện tại cho phép đơn giản hóa việc phân tích và khai thác như vậy, nhưng nhiệm vụ này vẫn đòi hỏi công sức rất lớn của các lập trình viên. Hơn nữa, chương trình có thể vẫn có lỗi khi cấu trúc của bất kỳ của các trang Web mà nó đọc bị thay đổi.

Trong Web ngữ nghĩa, trang sẽ được đánh dấu lên không chỉ với các thông tin chi tiết cần hiển thi; mà còn được đánh dấu dựa trên ý nghĩa của nội dung. Nói cách khác, như ví dụ trên thì trang kết quả chứa những gì giá là. Một chương trình sẽ trích xuất giá sẽ tìm thấy những giá ngay cả khi bố trí của trang đã được thay đổi. Nói cách khác, đánh dấu trên Web sẽ tiến triển từ các cú pháp chỉ đơn thuần là cấu trúc của các thông tin đến ngữ nghĩa để bắt ý nghĩa của các thông tin.

1.4 BÀI TẬP

Sử dụng một công cụ lập trình Web như Java Server Pages (JSP) hoặc Active Server Pages (ASP) để xây dựng một Website cho cửa hàng bán sách trực tuyến. Mỗi cuốn sách có các thông tin về tác giả, tựa đề sách, nhà xuất bản, năm xuất bản, và giá. Yêu cầu:

- Xây dựng cơ sở dữ liệu quản lý sách cho cửa hàng;
- Phát triển các trang Web cho phép người quản trị cập nhật, bổ sung, sửa, xóa, thống kê các quyển sách đã được bán và còn lại theo thời điểm nào đó;

- Phát triển các trang Web cho phép người dùng có thể tìm kiếm và đặt mua sách.

CHƯƠNG 2

CÁC CHUẨN CƠ BẢN CỦA DỊCH VỤ WEB

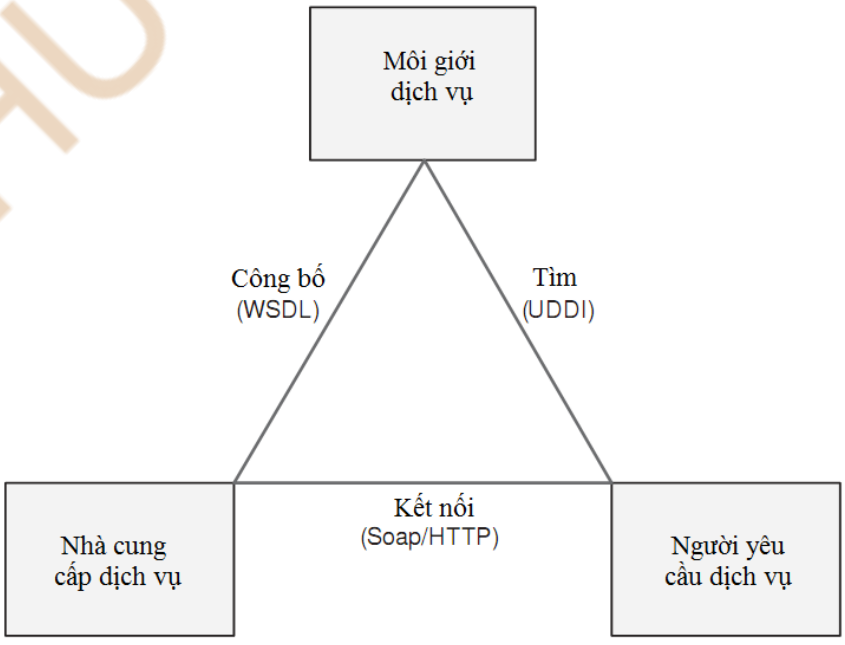
Mặc dù vấn đề dịch vụ Web đang được quan tâm; song ý tưởng của cung cấp các dịch vụ trên Web là khá cũ. Nhìn lại chúng ta có thể thấy rằng việc cung cấp và sử dụng dịch vụ Web đã có từ nhiều năm trước đây. Ví dụ, danh sách địa chỉ thư điện tử của những người phản hồi là những dịch vụ mà ta có thể đăng ký, hay những catalogue trực tuyến của các danh sách gửi thư với các chủ đề cụ thể mà ta đang quan tâm. Sự khác biệt chính giữa các dịch vụ cũ và các dịch vụ Web hiện đại là đối với các dịch vụ cũ thì cần thiết phải có sự can thiệp của con người. Ngày nay, các dịch vụ được mô tả và sử dụng theo các chuẩn. Mô hình kiến trúc chung cho các dịch vụ Web được thể hiện trong hình vẽ 2.1; Nó bao gồm ba đối tượng:

1. Nhà cung cấp dịch vụ (service provider): tạo ra các dịch vụ Web và quảng cáo cho người sử dụng tiềm năng đăng ký các dịch vụ Web với các nhà môi giới dịch vụ.

2. Môi giới dịch vụ (service broker): duy trì một đăng ký các dịch vụ xuất bản và giới thiệu các nhà cung cấp dịch vụ đến những người yêu cầu dịch vụ.

3- Người yêu cầu dịch vụ (service resquestor): người tìm kiếm, đăng ký và yêu cầu sử dụng dịch vụ phù hợp từ môi giới dịch vụ.

Các kiến trúc cho các dịch vụ Web được thành lập trên nguyên tắc và tiêu chuẩn để kết nối, truyền thông, mô tả, và khai phá. Đối với các nhà cung cấp và yêu cầu của các dịch vụ được thông tin kết nối và trao đổi thì cần thiết phải có một ngôn ngữ chung. Đó chính là ngôn ngữ đánh dấu mở rộng (eXtensible Markup Language hay XML). Một giao thức chung cũng cần thiết cho hệ thống để giao tiếp với nhau để người dùng có thể yêu cầu dịch vụ đó chính là giao thức truy cập đối tượng đơn giản (Simple Object Access Protocol, SOAP).



Hình 2.1: Mô hình kiến trúc chung cho các dịch vụ Web

Các dịch vụ phải được mô tả theo một định dạng máy có thể đọc được, mà tên của các hàm, các tham số, và các kết quả có thể được xác định. Điều này được cung cấp bởi ngôn ngữ mô tả dịch vụ Web (Web Services Description Language hay WSDL). Cuối cùng, người dùng và các doanh nghiệp cần có một cách để tìm các dịch vụ mà họ yêu cầu. Điều này có thể được giải quyết bởi chuẩn mô tả, khai phá, và tích hợp (Universal Description Discovery Integration hay UDDI). Bên cạnh các chuẩn XML, SOAP, WSDL, UDDI cần có các phương pháp cho đại diện ngữ nghĩa của các dịch vụ sẽ được trình bày ở các chương tiếp theo. Trong chương này ta sẽ tìm hiểu bốn chuẩn ngôn ngữ cơ bản cho dịch vụ Web là XML, SOAP, WSDL và UDDI.

2.1 NGÔN NGỮ ĐÁNH DẤU MỞ RỘNG XML

Ngôn ngữ cơ bản nhất của các ngôn ngữ trên là ngôn ngữ đánh dấu mở rộng XML. Các thẻ XML truyền đạt thông tin về ý nghĩa của dữ liệu. Không chỉ mô tả cách thức dữ liệu sẽ hiển thi như là trường hợp của HTML. XML tuân thủ cú pháp của HTML, vì vậy việc phân tích và xử lý trở nên dễ dàng hơn. XML có thể cung cấp các thẻ để mô tả tài liệu có cấu trúc hoặc phi cầu trúc. XML còn cho phép truy vấn dữ liệu cảm cấu trúc (structure-sensitive), có nghĩa là chúng ta có thể truy vấn một tài liệu XML dựa trên cấu trúc của nó theo lĩnh vực khác nhau. Ngoài ra, dữ liệu XML được gắn thẻ có thể được xác nhận một cách máy móc. Nói tóm lại, XML cung cấp một định dạng dữ liệu cho các tài liệu và dữ liệu có cấu trúc, nhưng không xác định ngữ nghĩa của các định dạng để chia sẻ thông tin và tri thức cũng như tương tác giữa các ứng dụng khác nhau.

Cú pháp XML cung cấp một tập luật để mô tả nội dung chứ không chỉ hiển thị nội dung, và nó được sử dụng để cung cấp cấu trúc cho dữ liệu. Một tài liệu XML tương ứng với một cấu trúc cây. Các phần tử chức năng (element function) được đặt vào trong các cặp dấu ngoặc. Ví dụ về một tài liệu XML đơn giản được trình bầy trong hình 1.1a.

```
<?xml version='1.0' encoding='UTF-8'?>
<temperature scale="Celsius">
  <value>25</value>
  <accuracy>2</accuracy>
  <ambience condition="shade"/>
</temperature>
```

Hình 1.1a: một tài liệu XML đơn giản

Trong ví dụ trên có thẻ mở là *<temperature>* và thẻ đóng là *</temperature>*. Các phần tử có thể chứa (nesting) dữ liệu văn bản hoặc chứa các phần tử khác. Các phần tử có thể có một số thuộc tính (attribute) mà mỗi thuộc tính gắn với một giá trị. Giá trị của thuộc tính phải là xâu kí tự (có thể được đóng gói vào cặp dấu ‘ ’ hoặc “ ”). Thuộc tính được đặt vào sau thẻ mở của phần tử (xem ví dụ trên ta thấy *<temperature scale=”Celsius”>*. Một tài liệu XML gồm một phần tử gốc (top-level element) có thể chứa các phần tử. Nói cách khác cây tài liệu có gốc là phần tử gốc.

2.2 GIAO THỨC TRUY NHẬP ĐỐI TƯỢNG ĐƠN GIẢN SOAP

Dự định ban đầu SOAP được phát triển để cung cấp cho các máy tính được nối mạng với các dịch vụ gọi thủ tục từ xa (Remote Procedure Call -RPC) được viết bằng XML. Nó đã trở thành một giao thức đơn giản và nhẹ để trao đổi thông điệp XML trên Web sử dụng HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer Protocol), và SIP (Session Initiation Protocol).

Trong thực tế, HTTP là giao thức phổ biến nhất cho SOAP và là lựa chọn cho các chuẩn tương thích, chẳng hạn như BP (Business Process) 1.0.

Thông điệp SOAP được có thể được chuyển từ một bên gửi đến bên nhận. Các thông điệp SOAP được định dạng như các tài liệu XML. Nói một cách khác SOAP không định ra các ngữ nghĩa ứng dụng hoặc cài đặt chi tiết mà nó chỉ cung cấp một cơ chế gọn nhẹ, hiệu quả cho phép trao đổi thông tin có cấu trúc và định dạng giữa các thành phần (components) trong môi trường phân tán bằng các tài liệu XML. Chính vì lẽ đó mà SOAP cho phép các ứng dụng chạy trên các nền tảng khác nhau có thể trao đổi thông tin cho nhau. Các đặc trưng của SOAP bao gồm:

- Tính đơn giản và dễ dàng mở rộng;
- Các thông điệp đều được định dạng XML;
- Giao thức truyền dữ liệu riêng;
- Kết nối giữa bên gửi và bên nhận là lỏng lẻo (loosed coupling); nghĩa là không cần cơ chế tham chiếu;
- Độc lập với nền tảng (platform independent) và độc lập với ngôn ngữ lập trình.

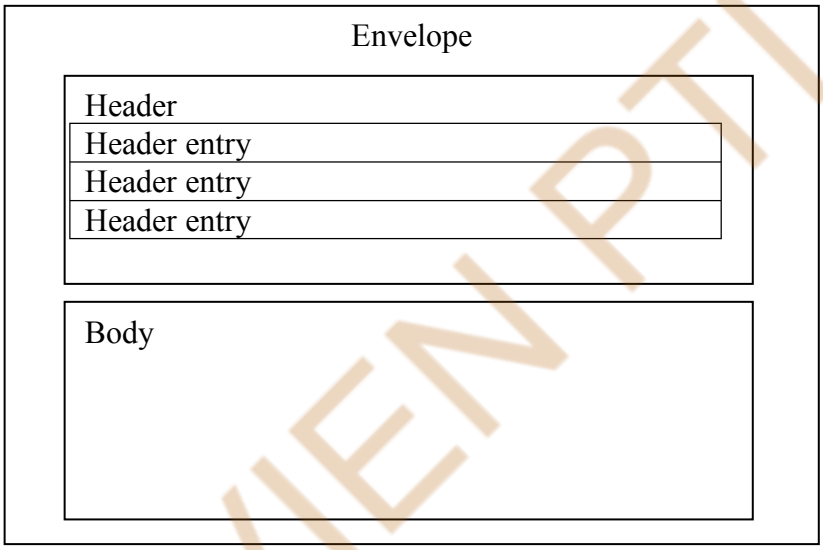
```
POST /temp HTTP/1.1
Host: www.socweather.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
SOAPAction: "http://www.socweather.com/temp"

<!— The above are HTTP header fields —>
<?xml version="1.0"?>
<env:Envelope
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
 env:encodingStyle=
   "http://schemas.xmlsoap.org/soap/encoding/"/>
 <env:Body>
  <m:GetTemp
   xmlns:m="http://www.socweather.com/temp.xsd">
    <m:City>Honolulu</m:City>
    <m:When>now</m:When>
  </m:GetTemp>
 </env:Body>
</env:Envelope>
```

Hình 2.2: Một yêu cầu SOAP

Hình 2.2 là ví dụ về một thông điệp SOAP. Trong ví dụ này, một công ty sản xuất có thể trực tiếp (ví dụ Dell) gọi một nhà cung cấp (ví dụ Intel) các danh mục hoặc đơn đặt hàng, hoặc có thể gửi một đơn đặt hàng đến các nhà cung cấp. Khi đó dịch vụ được mô hình hóa như các phương thức, sau đó thành phần dịch vụ được thực hiện thông qua các kịch bản (scripts) để gọi đến phương thức.

2.2.1 Cấu trúc và chức năng của SOAP

Thành phần gốc của một thông điệp SOAP là phong thư (envelop). Phong thư chứa các thành phần Header và Body. Trong Header có các thành phần con gọi là Header entry dùng để diễn giải các ngữ nghĩa cho thông điệp SOAP. Nhờ đó, thông điệp SOAP được định tuyến dựa trên các thông tin trong phần Header.



Hình 2.3: Cấu trúc thông điệp SOAP

Hình 2.4 cho thấy một thông điệp SOAP được đóng gói trong phương thức POST của giao thức HTTP. Phương thức POST của HTTP được sử dụng thay vì phương thức GET. Thông điệp được gửi đến www.socweather.com để yêu cầu các dịch vụ Web thực thi hàm GetTemp sử dụng "Honolulu" như giá trị của tham số City và "now" làm giá trị của tham số When. Các thông điệp kết quả SOAP chứa nhiệt độ là tham số DegreesCelsius với giá trị 30.

2.2.2 Body và Header

Mỗi thông điệp SOAP phải bao gồm phần body, thường được thông dịch (interpreted) bởi ultimateReceiver. Header cung cấp thông tin về các điểm đến trung hoặc cuối cùng của thông điệp SOAP (xem hình 2.2a).

```
<!ENTITY SOAPENV
    http://www.w3.org/2003/05/soap-envelope>
<env:Header>
  <c:converse
    xmlns:c='http://www.socweather.com/conversation.xsd'
    env:role='&SOAPENV;/role/next'>
   <c:msgID>
     uuid:0123456789-0123456789-0123456789
   </c:msgID>
  </c:converse>
</env:Header>
```

Hình 2.2a: ví dụ về SOAP header

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
SOAPAction: "http://www.socweather.com/temp"

<?xml version="1.0"?>
<env:Envelope
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
 env:encodingStyle=
   "http://schemas.xmlsoap.org/soap/encoding/"/>
 <env:Body>
   <m:GetTempResponse
      xmlns:m="http://www.socweather.com/temp.xsd">
       <DegreesCelcius>30</DegreesCelcius>
   </m:GetTempResponse>
 </env:Body>
</env:Envelope>
```

Hình 2.4: Một thông điệp SOAP

2.3 WSDL

2.3.1 Cấu trúc của WSDL

Mô hình kiến trúc cho các dịch vụ Web giả thiết là các dịch vụ có thể được tìm ra và sử dụng. Nghĩa là chúng ta sẽ có các mô tả về dịch vụ một cách chính xác. Ngôn ngữ mô tả dịch vụ Web *(*Web Services Description Language hay WSDL) là một ngôn ngữ XML dùng để mô tả một giao diện lập trình cho một dịch vụ Web [Christensen *et al.*, 2001]. Mô tả bao gồm định nghĩa của các kiểu dữ liệu, định dạng của thông điệp đầu vào và đầu ra, các hoạt động được cung cấp bởi dịch vụ (như GetTemp), địa chỉ mạng, và các ràng buộc giao thức. Để hiểu được WSDL một cách rõ ràng nhất, xem ví dụ ở hình 2.5 sau đây.

```
<?xml version ="1.0"?>
<!-- the root element , w s d l : d e f i n i t i o n s , d e f i n e s a s e t of -->
<!-- r e l a t e d s e r v i c e s -->
<w s d l : d e f i n i t i o n s name="Temperature"
targetNamespace="http://www.socweather.com/schema"
x m l n s : t s="http://www.socweather.com/TempSvc.wsdl"
x m l n s : t s x s d="http://schemas.socweather.com/TempSvc.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<!-- ws d l : t y p e s e n c a p s u l a t e s schema d e f i n i t i o n s of -->
<!-- communication t y p e s ; here using xsd -->
<w s d l : t y p e s>
<!-- a l l type d e c l a r a t i o n s are expressed in xsd -->
<xsd:schema
targetNamespace="http://namespaces.socweather.com"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<!-- xsd d e f : GetTemp [ Ci ty s t r i n g , When s t r i n g ] -->
<x s d : e l e m e n t name="GetTemp">
<xsd:complexType>
<xsd:sequence>
```

```
<x s d : e l e m e n t name="City" type="string"/>
<x s d : e l e m e n t name="When" type="string"/>
</ xsd:sequence>
</xsd:complexType>
</ xsd:element>
<!-- xsd d e f : GetTempResponse [ DegreesCel s ius i n t e g e r ] -->
<x s d : e l e m e n t name="GetTempResponse">
<!-- XML Schema e n t r y as above -->
</ xsd:element>
<!-- xsd d e f : GetTempFault [ errorMessage s t r i n g ] -->
<x s d : e l e m e n t name="GetTempFault">
<!-- XML Schema e n t r y as above -->
</ xsd:element>
</xsd:schema>
</ w s d l : t y p e s>
<!-- wsdl :message element s d e s c r i b e p o t e n t i a l t r a n s a c t i o n s -->
<!-- Most messages , as here , have only one par t . M u l t i p l e -->
<!-- par t s provide a way t o aggregate complex messages -->
<!-- r e q u e s t GetTempRequest i s of type GetTemp -->
<wsdl:message name="GetTempRequest">
<w s d l : p a r t name="body" element="tsxsd:GetTemp"/>
</ wsdl:message>
<!-- response GetTempResponse i s of t y p e GetTempResponse -->
<wsdl:message name="GetTempResponse">
<w s d l : p a r t name="body" element="tsxsd:GetTempResponse"/>
</ wsdl:message>
<!-- wsdl :por tType d e s c r ib e s messages in an o p e r a t i o n -->
<wsdl:portType name="GetTempPortType">
<!-- w s d l : o p e r a t i o n de s c r ibe s the e n t i r e pr o t o c o l from -->
<!-- input to o u t p u t or f a u l t -->
<w s d l : o p e r a t i o n name="GetTemp">
<!-- The order input preceding o u t p u t i n d i c a t e s the -->
<!-- request-response o p e r a t i o n t y p e -->
<w s d l : i n p u t message="ts:GetTempRequest"/>
<w s d l : o u t p u t message="ts:GetTempResponse"/>
<w s d l : f a u l t message="ts:GetTempFault"/>
</ w s d l : o p e r a t i o n >
</ wsdl:portType>
<!-- w s d l : b i n d i n g s p e c i f i e s a s e r i a l i z a t i o n pr o t o c o l -->
<w s d l : b i n d i n g name="TempSvcSoapBinding"
type="ts:GetTempPortType">
<!-- l e v e r a g e o f f s o a p : b i n d i n g document s t y l e -->
<s o a p : b i n d i n g s t y l e ="document"
t r a n s p o r t ="http://schemas.xmlsoap.org/soap/http"/>
<!-- semi-opaque c o n t a i n e r of network t r a n s p o r t d e t a i l s -->
<!-- c l a s s e d by soap:bi n d i n g above @@@-->
<w s d l : o p e r a t i o n name="GetTemp">
<s o a p : o p e r a t i o n
soapAction="http://www.socweather.com/TempSvc"/>
<!-- f u r t h e r s p e c i f y that the messages in the -->
<!-- w s d l : o p e r a t i o n "GetTemp" us e SOAP ? @@@ -->
<w s d l : i n p u t>
<soap:body use="literal"
namespace="http://schemas.socweather.com/TempSvc.xsd"/>
```

```
</ w s d l : i n p u t>
<!-- As above f o r ws d l : o u t p u t and ws d l : f a u l t -->
</ w s d l : o p e r a t i o n >
</ wsdl:binding>
<!-- w s d l : s e r v i c e names a new s e r v i c e "TemperatureService" -->
<w s d l : s e r v i c e name="TemperatureService">
<wsdl:documentation>socweather . com t e m p e r a t u r e s e r v i c e
</wsdl:documentation>
<!-- connect i t to the binding "TempSvcSoapBinding" above -->
<w s d l : p o r t name="GetTempPort" binding="ts:TempSvcSoapBinding">
<!-- g i v e t h e binding a network addr e s s -->
<s o a p : a d d r e s s l o c a t i o n ="http://www.socweather.com/TempSvc"/>
</ws d l : p o r t>
</ w s d l : s e r v i c e >
</ w s d l : d e f i n i t i o n s >
```

Hình 2.5: Một ví dụ WSDL

WSDL cho biết tên của các dịch vụ, như GetTemp, các loại thông số đầu vào, như String, các loại thông số đầu ra, như Integer, cấu trúc định nghĩa lược đồ XML (từ xSD namespace) cho các đầu vào và đầu ra, các hoạt động được cung cấp bởi dịch vụ như GetTemp, thứ tự giao thức của mỗi hoạt động từ đầu vào đến đầu ra hoặc lỗi. Thêm nữa giao thức serialization có có thể được dùng trong trao đổi thông tin như SOAP. Ngoài ra các địa chỉ mạng để có thể tìm thấy các dịch vụ được đặt trong khuôn dạng URL.

2.3.2 Chức năng của WSDL

WSDL mô tả bốn kiểu hoạt động, đặc trưng cho các hành vi của một đầu cuối. Chúng được xác định từ quan điểm của việc xây dựng nên các dịch vụ Web.

- One-way: Nhận một thông điệp.
- Notification: Gửi một thông điệp.
- Request-response: Nhận một yêu cầu và đưa ra một trả lời tương ứng.
- Solicit-response: Tạo ra một yêu cầu và nhận lại một trả lời tương ứng.

Các kiểu hoạt động có thể được thiết kế dựa trên các kiểu đơn hướng, nhưng chúng được xác định như trên vì đây là thể hiện của các mẫu thiết kế quan trọng. Ví dụ, nếu là máy chủ trong các lời gọi thủ tục từ xa (Remote Procedure Call hay RPC) thì tương ứng với Request-response, còn nếu là máy khách trong RPC thì tương ứng với Solicit-response. Như vậy, những kiểu hoạt động này đã mô tả trước các mẫu trao đổi thông điệp của SOAP. Trong số các kiểu trên, one-way và request–response là những hoạt động đang được sử dụng nhiều và chúng được hỗ trợ bởi HTTP cùng với các phương pháp lập trình hướng đối tượng phổ biến.

WSDL 2.0 có một tập các kiểu hoạt động phong phú hơn. Những hoạt động này bao gồm nhận hoặc gửi nhiều câu trả lời cho một truy vấn đơn. Tuy nhiên trong giới hạn ở đây sẽ không mô tả chi tiết về các hoạt động này.

2.3.3 Xây dựng WSDL

Việc xây dựng WSDL giúp phân chia đặc tả WSDL thành hai phần chính: phần giao diện (interface) và phần cài đặt (implementation). Việc phân chia đặc tả WSDL như thế này sẽ làm tăng khả năng mô đun hóa và phân tách được giao diện dịch vụ để từ đó có thể tái sử dụng và giúp tạo thêm nhiều phương pháp thực hiện.

Phần giao diện WSDL (interface) là thành phần trừu tượng hơn để mô tả một dịch vụ bằng cách bổ sung phần tử *definition* cho các phần tử con như types, import, message, portType và binding. Nghĩa là một giao diện có thể sử dụng các giao diện khác.

Để thực hiện dịch vụ WSDL cần phải xem xét đến các chi tiết cụ thể của việc gắn kết (binding) dịch vụ. Phần tử *definition* của dịch vụ phải có một phần tử *import* để kết nhập ít nhất một giao diện WSDL và một phần tử *service* (dịch vụ), trong đó bao gồm các phần tử *port* (cổng). Phần tử *import* xác định một định danh và vị trí cho *namespace* (không gian tên) được kết nhập.

2.4 UDDI (Universal Description, Discovery, and Integration)

Đặc tả UDDI *(Universal Description, Discovery, and Integration)* [UDDI, 2000] mô tả cơ chế đăng ký và định vị các dịch vụ Web. Đặc tả này định nghĩa một *registry* (cơ sở dữ liệu đăng ký) trực tuyến để các tổ chức (ví dụ các nhà cung cấp dịch vụ) có thể mô tả về tổ chức và đăng ký dịch vụ Web của họ. Registry sau đó có thể được sử dụng bởi các bên yêu cầu dịch vụ và người sử dụng để xác định vị trí các dịch vụ mà họ cần. UDDI giúp các nhà cung cấp dịch vụ liên kết các dịch vụ của họ với nhau và giúp các bên yêu cầu trong việc khám phá (tìm ra) các dịch vụ. Đây là một điều kiện tiên quyết cho việc tạo ra các dịch vụ Web.

2.4.1 Cấu trúc của UDDI

Một UDDI registry bao gồm 3 thành phần:

- Trang trắng (White Pages) — địa chỉ, thông tin liên hệ, và các định danh đã biết;
- Trang vàng (Yellow Pages) — phân loại dựa trên nguyên tắc phân loại tiêu chuẩn;
- Trang xanh (Green Pages) — thông tin kỹ thuật về các dịch vụ đưa ra của các doanh nghiệp.

Trang trắng cung cấp thông tin về các doanh nghiệp cung cấp dịch vụ. Thông tin này bao gồm tên của doanh nghiệp và một mô tả về doanh nghiệp - có thể bằng nhiều ngôn ngữ. Sử dụng thông tin này có thể giúp tìm thấy một dịch vụ khi đã biết một số thông tin (ví dụ, xác định một dịch vụ dựa theo tên của nhà cung cấp).

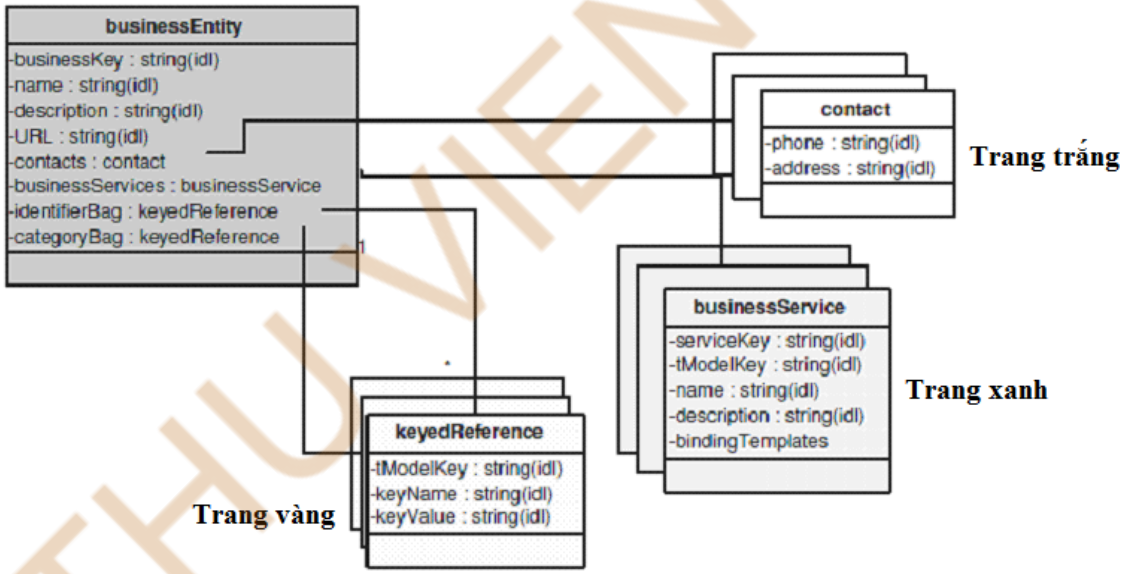
Thông tin liên lạc cho các doanh nghiệp cũng được cung cấp - ví dụ địa chỉ doanh nghiệp, số điện thoại và các thông tin khác.

Các trang vàng được thực hiện theo các cặp tên-giá trị nhằm cho phép bất kỳ định danh phân loại hợp lệ nào đều được gắn vào các trang trắng cho một doanh nghiệp. Việc tìm kiếm trong một trang vàng có thể được thực hiện để xác định loại ngành công nghiệp hoặc loại sản phẩm cụ thể của doanh nghiệp, hoặc xác định vị trí địa lý của doanh nghiệp.

Các trang xanh chứa những thông tin doanh nghiệp sử dụng để mô tả cách các doanh nghiệp khác có thể tiến hành thương mại điện tử với họ. Thông tin trang xanh là một mô hình lồng nhau bao gồm các quy trình kinh doanh, giới thiệu dịch vụ và thông tin kết nối. Các thông tin không phụ thuộc vào ngôn ngữ, nền tảng và cách cài đặt. Các dịch vụ cũng có thể được phân loại.

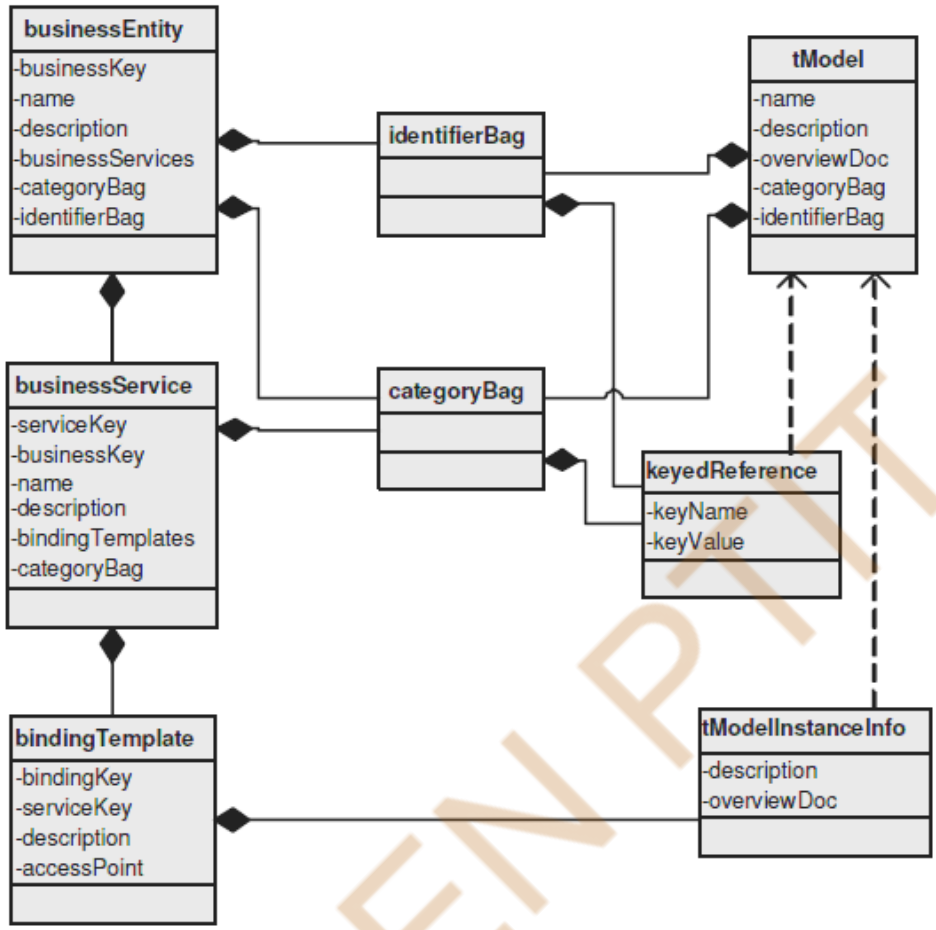
2.4.2 Chức năng của UDDI

UDDI chính là một dịch vụ Web dựa trên XML và SOAP. Ví dụ, một bản đăng ký kinh doanh là một tài liệu XML. Khách hàng sử dụng một tập các giao diện SOAP (SOAP interfaces) được xác định trước để tìm kiếm đăng ký cho một dịch vụ web mong muốn. Nhà cung cấp sử dụng giao diện SOAP để đăng ký hai loại thông tin: (1) mô hình kỹ thuật (tModel), là các giao thức dịch vụ trừu tượng mô tả hành vi của một dịch vụ Web riêng lẻ, và (2) các thực thể kinh doanh (businessEntity), trong đó mô tả một cài đặt dịch vụ và cung cấp mô tả về các đặc tả của nhiều tModels. Lưu ý là mỗi đặc tả, vận chuyển, giao thức, hoặc không gian tên riêng biệt được trình bày bởi một tModel. Tuy nhiên, một UDDI registry không thực sự lưu trữ các đặc tả và chi tiết như vậy. Một UDDI tModel chỉ đơn giản chứa các địa chỉ (URL) nơi chứa những tài liệu kỹ thuật, siêu dữ liệu về các tài liệu và một khóa để nhận biết tModel.

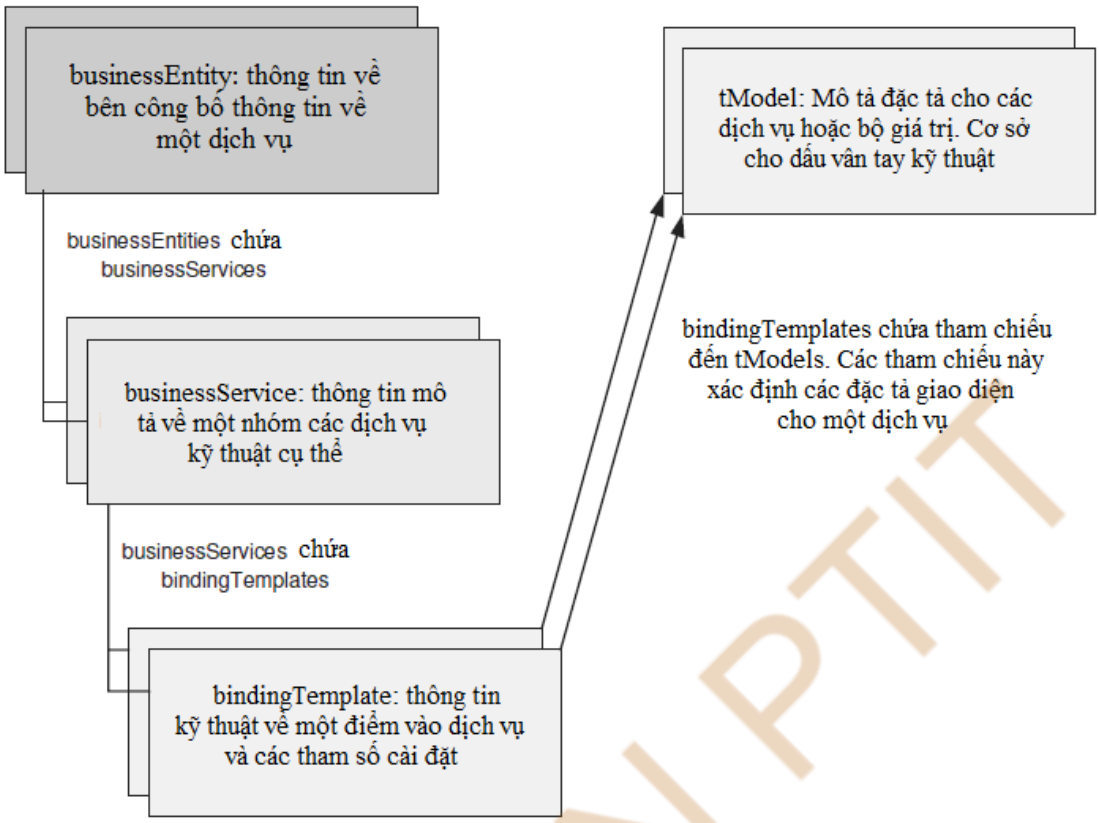


Hình 2.6: Các trang biểu diễn một thực thể doanh nghiệp trong UDDI registry

Hình 2.6 cho thấy các trang vàng, trắng và xanh cho một doanh nghiệp. Một businessEntity là cấu trúc mức cao nhất cho tất cả các thông tin liên quan đến một doanh nghiệp, được thể hiện rõ hơn trong hình 2.7. Các thành phần chính của một businessEntity UDDI và các mối quan hệ giữa chúng được thể hiện trong hình 2.8.

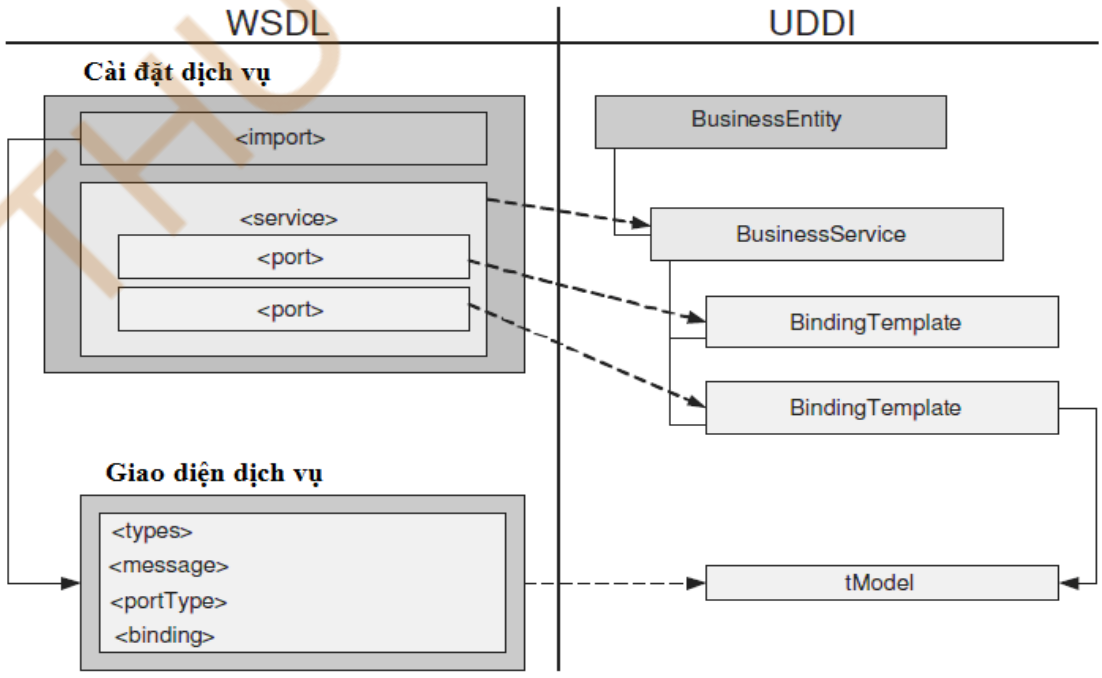


Hình 2.7: Mô hình thông tin UML cho một thực thể doanh nghiệp trong UDDI registry



Hình 2.8: Các cấu trúc dữ liệu lõi và các mối quan hệ giữ chúng cho một thực thể doanh nghiệp UDDI

Trong nội dung này sẽ chỉ quan tâm chủ yếu đến việc đăng ký các dịch vụ Web, vì vậy chúng ta sẽ chỉ ánh xạ mô tả WSDL của dịch vụ Web tới các mô tả dịch vụ UDDI. Hình 2.9 cho thấy sự tương ứng giữa các trường của một mô tả WSDL và các trường của một UDDI businessService.



Hình 2.9: So sánh giữa một tài liệu WSDL và một tài liệu đăng ký UDDI

2.4.3 Xây dựng UDDI

UDDI xác định hai hàm giao diện lập trình ứng dụng (Application Programming Interface API) cho việc truy cập đến một UDDI registry: Inquiry API để lấy thông tin từ một registry và API Publish dùng để lưu trữ thông tin đó. Publish API yêu cầu truy nhập được xác thực (việc này khá đặc biệt cho một registry và không theo quy định của UDDI) nhưng Inquiry API thì không. Các API hiện hỗ trợ 28 thông điệp SOAP, sau đây là một số thông điệp quan trọng:

- Inquiry API
  - ✓ Tìm một doanh nghiệp hoặc dịch vụ của doanh nghiệp và các đặc tính của nó
  - ✓ Lấy thông tin chi tiết cần để tương tác với một doanh nghiệp
- Publishing API
  - ✓ Lưu lại thông tin về một doanh nghiệp hoặc dịch vụ của doanh nghiệp
  - ✓ Xóa
  - ✓ An toàn

2.4.3.1 Đăng ký và công bố một dịch vụ

Sau khi đã tìm hiểu các thành phần cơ bản của một UDDI, chúng ta sẽ xem xét một ví dụ đăng ký từ công ty WeatherService và cách thức để dịch vụ "thông báo nhiệt độ hiện thời" của công ty có thể được tìm thấy và sau đó được sử dụng bởi khách hàng. Đầu tiên, công ty WeatherService sẽ trao đổi hai thông điệp SOAP với một UDDI registry (có thể registry này được duy trì bởi IBM tại https://uddi.ibm.com/ubr). Thông điệp SOAP đầu tiên sẽ gọi hoạt động getauthToken để thiết lập xác thực. Tiếp theo, như trong hình 2.10, sẽ đăng ký WeatherService là một thực thể kinh doanh.

```
POST / HTTP / 1 . 1
Host: www. socweather . com
Content−Type: t e x t / xml ; c h a r s e t ="utf-8"
Content−Length: nnnn
SOAPAction: ""
<?xml version ="1.0" encoding="UTF-8" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
<s a v e b u s i n e s s xmlns="urn:uddi-org:api_v3">
<b u s i n e s s D e t a i l t r u n c a t e d ="false">
<b u s i n e s s E n t i t y businessKey="...K1...">
<discoveryURLs>
<discoveryURL useType="homepage">
h t t p : / /www. socweather . com/ WeatherService . html
</discoveryURL>
</discoveryURLs>
<name xml:lang="en">WeatherService Inc . </name>
<d e s c r i p t i o n xml:lang="en">P r o v i d e r o f t e m p e r a t u r e s e r v i c e s
</ de s c r i p t i o n >
<c o n t a c t s >
<c o n t a c t>
```

```
<d e s c r i p t i o n xml:lang="en">P r e s i d e n t </ de s c r i p t i o n >
<personName>Hot N. Cold</personName>
<phone useType="Voice">803−555−1234</phone>
</ c o n t a c t>
</ c o n t a c t s >
<i d e n t i f i e r B a g >
<keyedReference
tModelKey="uddi:uddi.org:ubr:identifier:dnb.com:d-u-n-s"
keyName="DUNS: WS Inc." keyValue="12-123-1234"/>
</ i d e n t i f i e r B a g >
<categoryBag>
<!−− NAICS C l a s s i f i c a t i o n −−>
<keyedReference tModelKey="uuid:K6"
keyName="Meterological services" keyValue="541990"/>
<!−− ISO 3166 Geographic Taxonomy −−>
<keyedReference tModelKey="uuid:K7"
keyName="North Carolina , USA" keyValue="US-NC"/>
</ categoryBag>
<b u s i n e s s S e r v i c e s >
<b u s i n e s s S e r v i c e serviceKey="...K2..." businessKey="...K1...">
<name xml:lang="en">Temperature Se r v i c e </name>
<d e s c r i p t i o n xml:lang="en">
Given a t ime and c i t y , i t r e t u r n s a temp e r a t u r e
</ d e s c r i p t i o n >
<bindingTemplates>
<bindingTemplate bindingKey="...K3..." serviceKey="...K2...">
<d e s c r i p t i o n xml:lang="en">
This s e r v i c e uses a SOAP/RPC encoded endpoint
</ de s c r i p t i o n >
<a c c e s s P o i n t URLType="http">
h t t p : / /www. s o cwe a th e r . com/ TempSvc
</ a c c e s s P o i n t>
<t M o d e l I n s t a n c e D e t a i l s >
<t M o d e l I n s t a n c e I n f o tModelKey="uuid:...K4..."/>
</ t M o d e l I n s t a n c e D e t a i l s >
</ bindingTemplate>
</ bindingTemplates>
</ b u s i n e s s S e r v i c e >
</ bu s i n e s s S e r v i c e s >
</ b u s i n e s s E n t i t y >
</ bu s i n e s s D e t a i l >
<tModelDetai l t r u n c a t e d ="false">
<tModel tModelKey="uuid:...K4...">
<name>TempSvc S p e c i f i c a t i o n </name>
<d e s c r i p t i o n xml:lang="en">tModel for s e r v i c e i n t e r f a c e d e f i n i t i o n
</ de s c r i p t i o n >
<overviewDoc>
<overviewURL>h t t p : / /www. s o cwe a th e r . com/ TempSvc . wsdl
</overviewURL>
</overviewDoc>
<categoryBag>
<keyedReference tModelKey="uuid:...K6...."
keyName="uddi-org:types" keyValue="wsdlSpec"/>
</ categoryBag>
```

```
</ tModel>
</ tModelDetail>
</ s a v e b u s i n e s s >
</env:Body>
</ env:Envelope>
```

Hình 2.10: SOAP body của một ví dụ về UDDI registry cho một thực thể kinh doanh

Hãy xem một số trường trong registry trên cho công ty WeatherService. Đầu tiên, gửi tới registry lệnh save_business, với thuộc tính xác định là đang sử dụng phiên bản 3 của UDDI. Một thông điệp save_business chứa một businessDetail và một số bất kỳ tModels. businessDetail chỉ ra nơi có thể tìm thấy trang web của WeatherService, cách thức có thể gọi cho chủ tịch công ty, v.v.

Tiếp tục xem mô tả về việc đăng ký, chỉ có một trong những dịch vụ được cung cấp bởi WeatherService được liệt kê là TemperatureService. Phần chi tiết được đưa ra trong một bindingTemplate, gồm hai phần chính: (1) URL chính xác để có thể truy cập dịch vụ và (2) tModel cung cấp thông tin truy nhập. tModel, trong trường hợp này có một khóa nhận dạng trỏ tới tModel, được đưa ra làm một phần của việc đăng ký, nhưng tModel có thể được quy định trong một thông điệp riêng biệt để đăng ký hoặc thậm chí có thể là một phần đăng ký của doanh nghiệp khác. tModels trong trường hợp này giống như các kiểu dữ liệu toàn cầu có thể tái sử dụng và định nghĩa bởi người dùng.

tModel trong thông điệp đăng ký xác định rằng TempSvc được định nghĩa trong WSDL và cung cấp một con trỏ đến các tài liệu WSDL thích hợp.

Tiếp theo, WeatherService gửi ba thông điệp SOAP tới UDDI registry để đăng ký ba tModels sau đây mô tả các hành vi của các dịch vụ nhiệt độ của công ty. Các hành vi được mô tả theo portType cho dịch vụ và các kết nối giao thức. Phần tử overview_Doc trong hai thông điệp đầu tiên (các hình 2.11 và 2.12) có chứa một phần tử overviewURL, trong đó có các URI cho giao diện WSDL của dịch vụ đang được công bố. Các liệt kê này mô tả việc đăng ký về tModels để có thể được sử dụng làm một phần của các dịch vụ khác, không chỉ là những dịch vụ của công ty WeatherService. Một lần nữa lưu ý rằng registry không lưu trữ tài liệu WSDL, mà chỉ chứa một con trỏ đến nó.

```
<save tModel xmlns="urn:uddi-org:api_v3">
<tModel tModelKey="uuid:...KA..." >
<name>GetTempPortType</name>
<overviewDoc>
<overviewURL>h t t p : / /www. s o cwe a t he r . com/ TempSvc . wsdl</overviewURL>
</overviewDoc>
<categoryBag>
<keyedReference tModelKey="uuid:...KB..."
keyName="portType namespace"
keyValue="http://www.socweather.com/TempSvc"/>
<keyedReference tModelKey="uuid:...KC..."
keyName="WSDL type" keyValue="portType"/>
</ categoryBag>
</ tModel>
</ save tModel>
```

Hình 2.11: tModel đầu tiên trong 3 tModel của công ty WeatherService xác định thông tin port (cổng)

```
<save tModel xmlns="urn:uddi-org:api_v3">
<tModel tModelKey="uuid:...KD...">
<name>TempSvcSoapBinding</name>
<overviewDoc>
<overviewURL>h t t p : / /www. s o cwe a t he r . com/ TempSvc . wsdl</overviewURL>
</overviewDoc>
<categoryBag>
<keyedReference tModelKey="uuid:...KE..."
keyName="binding namespace"
keyValue="http://www.socweather.com/TempSvc"/>
<keyedReference tModelKey="uuid:...KF..."
keyName="WSDL type" keyValue="binding"/>
<keyedReference tModelKey="uuid:...KG..."
keyName="portType reference" keyValue="uuid:...KH..."/>
<keyedReference tModelKey="uuid:...KI..."
keyName="SOAP protocol" keyValue="uuid:...KJ..."/>
<keyedReference tModelKey="uuid:...KK..."
keyName="HTTP transport" keyValue="uuid:...KL..."/>
<keyedReference tModelKey="uuid:...KM..."
keyName="uddi-org:types" keyValue="wsdlSpec"/>
</ categoryBag>
</ tModel>
</ save tModel>
```

Hình 2.12: tModel thứ hai trong 3 tModel của công ty WeatherService xác định ràng buộc giao thức

Hình 2.13 là một mô tả bổ sung của businessService của công ty WeatherService, trong đó chứa các chi tiết về port (cổng) của dịch vụ Temperature Service.

```
<s a v e s e r v i c e xmlns="urn:uddi-org:api_v3">
<b u s i n e s s S e r v i c e serviceKey="...K2..." businessKey="...K1...">
<name>Temperature Se r v i c e </name>
<bindingTemplates>
<bindingTemplate bindingKey="...KP..." serviceKey="...KN...">
<a c c e s s P o i n t URLType="http"> h t t p : / /www. s o cwe a t he r . com/ TempSvc
</ a c c e s s P o i n t>
<t M o d e l I n s t a n c e D e t a i l s >
<t M o d e l I n s t a n c e I n f o tModelKey="uuid:...KQ...">
<d e s c r i p t i o n xml:lang="en">The w s d l : b i n d i n g the w s d l : p o r t
implements ; ins t anc ePa rms s p e c i f i e s the por t l o c a l name .
</ d e s c r i p t i o n >
<i n s t a n c e D e t a i l s >
<instanceParms>GetTempPort</ instanceParms>
</ i n s t a n c e D e t a i l s >
</ tModelInstanceInfo>
<t M o d e l I n s t a n c e I n f o tModelKey="uuid:...KR...">
<d e s c r i p t i o n xml:lang="en">
The wsdl :por tType t h a t t h i s wsdl : p o r t implements .
</ d e s c r i p t i o n >
</ tModelInstanceInfo>
</ t M o d e l I n s t a n c e D e t a i l s >
</ bindingTemplate>
</ bindingTemplates>
<categoryBag>
<keyedReference tModelKey="uuid:...KS..."
keyName="WSDL type" keyValue="service"/>
<keyedReference tModelKey="uuid:...KT..."
keyName="service namespace"
keyValue="http://www.socweather.com"/>
<keyedReference tModelKey="uuid:...KU..."
keyName="service local name" keyValue="TemperatureService"/>
</ categoryBag>
</ b u s i n e s s S e r v i c e >
</ s ave s e r v i c e >
```

Hình 2.13: tModel thứ ba trong 3 tModel của công ty WeatherService cập nhật dịch vụ đang cung cấp

Như có thể thấy từ các ví dụ trên, các mục UDDI là các tài liệu XML để bổ sung cho WSDL bằng cách xác định các cổng, giao diện và các ràng buộc giao thức của một dịch vụ. Cũng lưu ý rằng UDDI luôn mở cho bất kỳ đăng ký loại hình dịch vụ nào, chứ không phải chỉ các dịch vụ Web dựa trên WSDL.

2.4.3.2 Tìm kiếm một dịch vụ

Khi đã đăng ký, các dịch vụ của WeatherService có thể được tìm thấy bởi một ứng dụng máy khách thông qua việc gửi các tài liệu XML trong Hình 2.14 đến registry theo dạng nội dung của một thông điệp SOAP. Lưu ý là mọi yêu cầu không cần phải chứng thực.

```
<?xml version ="1.0" encoding="UTF-8"?>
<f i n d b u s i n e s s xmlns="urn:uddi-org:api_v3">
<f i n d Q u a l i f i e r s >
<f i n d Q u a l i f i e r >uddi :uddi . o r g : f i n d q u a l i f i e r : e x a c t m a t c h
</ f i n d Q u a l i f i e r >
</ f i n d Q u a l i f i e r s >
<!--f i n d i n f o about a l l bus ine s s e s named "WeatherService Inc." -->
<name>WeatherService Inc . </name>
</ f i n d b u s i n e s s >
```

Hình 2.14: Một yêu cầu ví dụ từ một máy khách nhằm xác định thông tin về WeatherService đã được lưu trong một UDDI registry

Các thông tin tổng hợp về công ty WeatherService được hiển thị trong hình 2.15. Phần quan trọng của danh sách này là businessKey và serviceKey, có thể được dùng trong các yêu cầu tiếp theo để tìm thêm thông tin về dịch vụ.

```
<?xml version ="1.0" encoding="UTF-8"?>
<b u s i n e s s L i s t >
<b u s i n e s s I n f o s >
<b u s i n e s s I n f o businessKey="...KO...">
<name>WeatherService , Inc . </name>
<s e r v i c e I n f o s >
<s e r v i c e I n f o serviceKey="...KN..." businessKey="...K1...">
<name>Temperature Se r v i c e </name>
</ s e r v i c e I n f o >
</ s e r v i c e I n f o s >
</ b u s i n e s s I n f o >
</ b u s i n e s s I n f o s >
</ b u s i n e s s L i s t >
```

Hình 2.15: Kết quả của một truy vấn để xác định thông tin về công ty WeatherService

2.5 BÀI TẬP

1. Hãy tìm phần tử, thuộc tính, giá trị thuộc tính trong đoạn tài liệu XML sau đây:

   <alpha delta=”gamma”>

   Beta

   </alpha>

2. Hãy viết một chương trình (bằng ngôn ngữ lập trình do bạn tự lựa chọn) với giao diện là một biểu mẫu Web với đầu vào là một lược đồ XML (schema). Khi người sử dụng nhập dữ liệu vào các trường dữ liệu trên biểu mẫu thì chương trình sẽ biến đổi dữ liệu vừa nhập thành dạng XML tương ứng với lược đồ XML đầu vào là lưu vào một file.
3. Hãy xem xét một phong thư SOAP sau (hình 2.16):

```
<!ENTITY SOAPENC
          "http://www.w3.org/2001/06/soap-encoding">
<s:Envelope xmlns:s="...">
 <s:Body>
  <order xmlns="..."
   s:encodingStyle=&SOAPENC;>
   <partName xsi:type="string">valve</partName>
   <quantity xsi:type="string">12</quantity>
  </order>
 </s:Body>
</s:Envelope>
```

Hình 2.16: Phong thư SOAP

Tương ứng với phương thức chữ ký nào trong Java:

A. string order (String partName, String quantity);

B. void order (String quantity, String partName);

C. void order (String quantity, Integer partName);

D. void partName (String s); void quantity(Integer n);

E. không có phương thức nào ở trên.

4. Thành phần nào dưới của WSDL không được hỗ trợ bởi điểm cuối:

   A. Multicast;

   B. one-way;

   C. request-response;

   D. solicit-response;

   E. notification.

5. Xây dựng một tài liệu WSDL mô tả một dịch vụ thông báo chứng khoán. Xác định các loại thông báo sau: loginRequest, logReply, stockQuoteRequest, stockQuoteResponse, và logoutRequest, và các hoạt động sau đây: QuoteToUser, LogIn, ProvideQuote, LogOut, và QueryNYSE.

6. Xây dựng một mô tả WSDL cho một dịch vụ danh bạ điện thoại (phone book) với hai thao tác GetPhoneNumber và SetPhoneNumber. Thao tác (operation) GetPhoneNumber nhận vào một tham số tên kiểu String và trả về một số điện thoại kiểu string; Ngược lại, SetPhoneNumber nhận vào một tham số tên kiểu String và một số điện thoại kiểu String và không trả về giá trị nào.

7. Giao thức UDDI được sử dụng vào việc gì:

   A. Tìm các dịch vụ SOAP thực thi một giao diện;

B. Mô tả giao diện của một dịch vụ SOAP;
C. Giao tiếp giữa SOAP và .NET;
D. Mô tả giao thức truyền thông của dịch vụ SOAP;
E. Mô tả một dịch vụ SOAP được triển khai trên một máy chủ Web như thế nào.

8. Một tModel UDDI là:

A. Một cách mô tả kinh doanh đa dạng, dịch vụ và các cấu trúc khuôn dạng được lưu trong UDDI registry;

B. Một mô tả kỹ thuật của các dịch vụ Web được hiển thị bởi cấu trúc dịch vụ kinh doanh;

C. Mô tả bằng tiếng Anh của nghiệp vụ;

D. Một định nghĩa mẫu thông điệp request-response;

E. Một mô hình chuyển dịch.

9. Một UDDI registry có thể được truy cập bởi giao diện SOAP nào dưới đây:

A. InquireSOAP và PublishSOAP;

B. UDDISOAPInterface;

C. SOAP với HTTP;

D. Publish và Query;

E. uddiSOAP

10. Bước đầu tiên để dịch một file WSDL để được sử dụng với UDDI là:

A. Chia thành 2 files: một file chứa phần giao diện (interface) và file còn lại chứa phần mô tả thực thi (implementation description).

B. Đăng kí file WSDL như một UDDI tModel

C. Thay đổi tất cả kiểu từ lược đồ XML thành lược đồ UDDI

D. Gửi nó đến một UDDI registry và thu nhận (gather) các thông điệp báo lỗi.

E. Viết lại nó sử dụng lược đồ UDDI.

11. Giả sử bạn phụ trách quản lý một UDDI registry có tên là MyUDDI.com và được đặt tại http://www.MyUDDI.com. Giả sử có người gửi một yêu cầu find_tModel trong một thông điệp SOAP, và registry của bạn sẽ trả về một tModel tương ứng. Registry của bạn cung cấp dịch vụ và bó có thể được mô tả bằng tài liệu WSDL. Bạn hãy viết file mô tả WSDL cho dịch vụ yêu cầu find_tModel cho registry của bạn.

12. * Các dịch vụ Web hiện tại được thực thi bằng việc sử dụng WSDL và SOAP có thể chạy tốt khi một người yêu cầu cần một biến thể hiện (instance) của dịch vụ cho một tương tác. Tuy nhiên, việc tương tác có thể phức tạp hơn khi một người mua (người yêu cầu) muốn mua một dịch vụ từ phía người bán (nhà cung cấp dịch vụ). Trong

trường hợp này người mua sẽ hỏi người bán báo giá cả của dịch vụ. Sau khi nhận được báo giá người mua sẽ đặt lệnh mua và người bán sẽ chấp nhận lệnh mua này. Người mua sau đó có thể truy nhập được dịch vụ từ người bán. Các biểu mẫu tương tác của quá trình này cần được mở rộng. Các chuẩn cơ bản của dịch vụ Web như UDDI, WSDL, SOAP, và XML cần được sử dụng hay thay đổi ra sao để thích hợp với các tương tác như trên.

CHƯƠNG 3
CÔNG NGHỆ CHO PHÁT TRIỂN DỊCH VỤ WEB

3.1 NỀN TẢNG CHO PHÁT TRIỂN DỊCH VỤ WEB

3.1.1 J2EE

Nền tảng Java 2 (Java 2 Enterprise Edition hay J2EE) xây dựng trên ngôn ngữ lập trình Java dùng để cung cấp một framework (khung làm việc) cho việc phát triển và triển khai các ứng dụng Java xoay quanh một máy chủ ứng dụng. Tuân theo những nguyên tắc chung của kiến trúc nhiều tầng, J2EE phân thành ba lớp chính:

1. Trình diễn (Presentation) - để tương tác với người sử dụng, thường là thông qua các ứng dụng máy khách.

2. Các đối tượng nghiệp vụ (Business objects) - để lưu giữ logic nghiệp vụ, bản chất là các ứng dụng Java hỗ trợ. Đây là phần cốt lõi của J2EE và chủ yếu chứa Java Beans (Enterprise Java Bean hay EJB) là các đối tượng phân tán được định nghĩa theo các luật. Các đối tượng này tự mô tả về các phương pháp để bên ngoài có thể gọi vào. EJB được chia làm hai loại, bao gồm phiên người dùng và thông tin cho các đối tượng.

EJB tạo điều kiện phát triển các ứng dụng có thể bảo trì bởi vì chúng lưu giữ logic nghiệp vụ của ứng dụng theo cách độc lập với việc triển khai. Nói cách khác, khi một ứng dụng được thiết kế sử dụng EJB, các quyết định về cách thức phân bổ tài nguyên (resources) cho các EJB chưa cần phải thực hiện ngay. Nếu một EJB nào đó thấy có nhu cầu sử dụng tài nguyên cao, nó có thể được triển khai và được sử dụng tài nguyên bổ sung, ví dụ như sử dụng một số lượng lớn các luồng (thread).

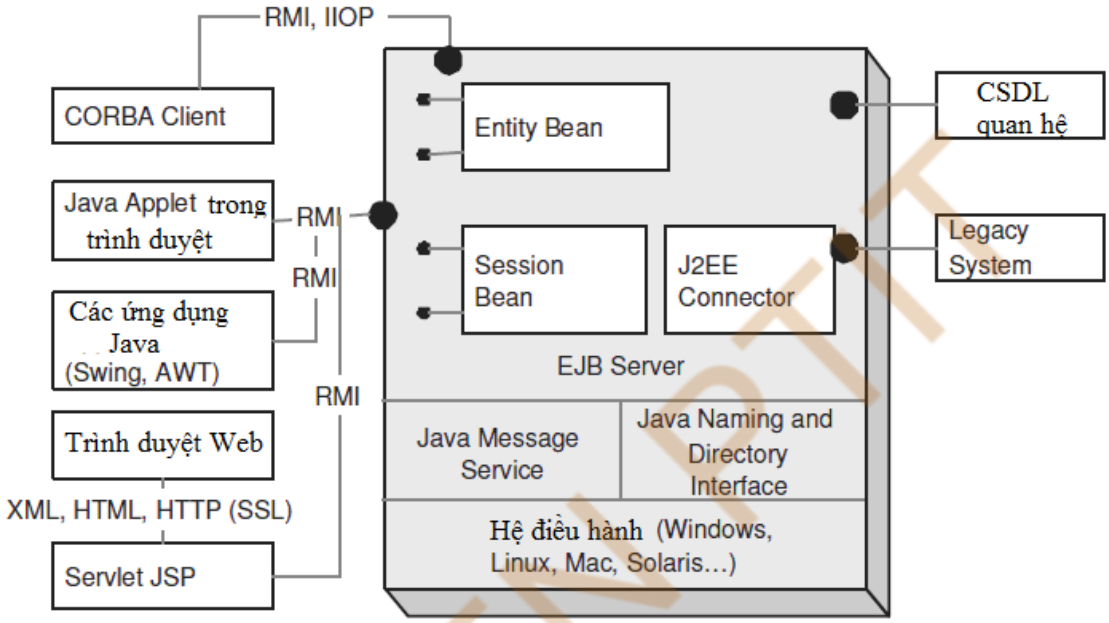
3. Backend - xử lý các tương tác với cơ sở dữ liệu, hệ thống quan trị nguồn lực doanh nghiệp (Enterprise Resources Planning hay ERP) và các hệ thống khác (ngay cả những thứ không có trong J2EE). Lớp này gắn kết chủ yếu với kiến trúc kết nối J2EE (J2EE Connector Architecture hay JCA) dùng cho việc hỗ trợ truyền thông đồng bộ và không đồng bộ giữa các hệ thống.

Máy chủ ứng dụng J2EE bao gồm một đối tượng vật chứa Web (Web container) và một đối tượng vật chứa EJB (EJB container). Đối tượng vật chứa Web lưu và chạy các servlet và các trang Java Server Pages (JSP). Nó gọi các Enterprise bean trong đối tượng vật chứa EJB. Máy chủ ứng dụng cũng tương tác với bên ngoài thông qua hàng đợi thông điệp và cơ sở dữ liệu. Đối tượng vật chứa EJB cung cấp dịch vụ mức hệ thống bao gồm cả kết nối, bảo mật, và các giao dịch.

J2EE cũng hỗ trợ cho hàng đợi thông điệp (messaging queue) nhờ sử dụng dịch vụ thông điệp Java (Java Message Service hay JMS). Thông điệp có thể được gửi và nhận, được đồng bộ bởi các client và các EJB. MDB (Message-Driven Beans) nhận và xử lý thông điệp từ hàng đợi theo cách không đồng bộ. Thông điệp gửi và nhận có thể là một phần của một giao dịch phân tán.

Giao diện lập trình đặt tên Java và giao diện danh bạ (Java Naming and Directory Interface hay JNDI) là một giao diện lập trình cho các registry siêu dữ liệu. Một ứng dụng

J2EE hoặc EJB có thể tự xác định vị trí các tài nguyên và các thành phần khác, do đó có thể phân tách được chi tiết cấu hình ra khỏi việc cài đặt. Chuẩn kết nối cơ sở dữ liệu Java (Java Database Connectivity hay JDBC) cung cấp các thư viện để mở và duy trì kết nối với cơ sở dữ liệu quan hệ, truy xuất siêu dữ liệu quan hệ, chuẩn bị và gửi các câu lệnh truy vấn SQL, và xử lý các kết quả.



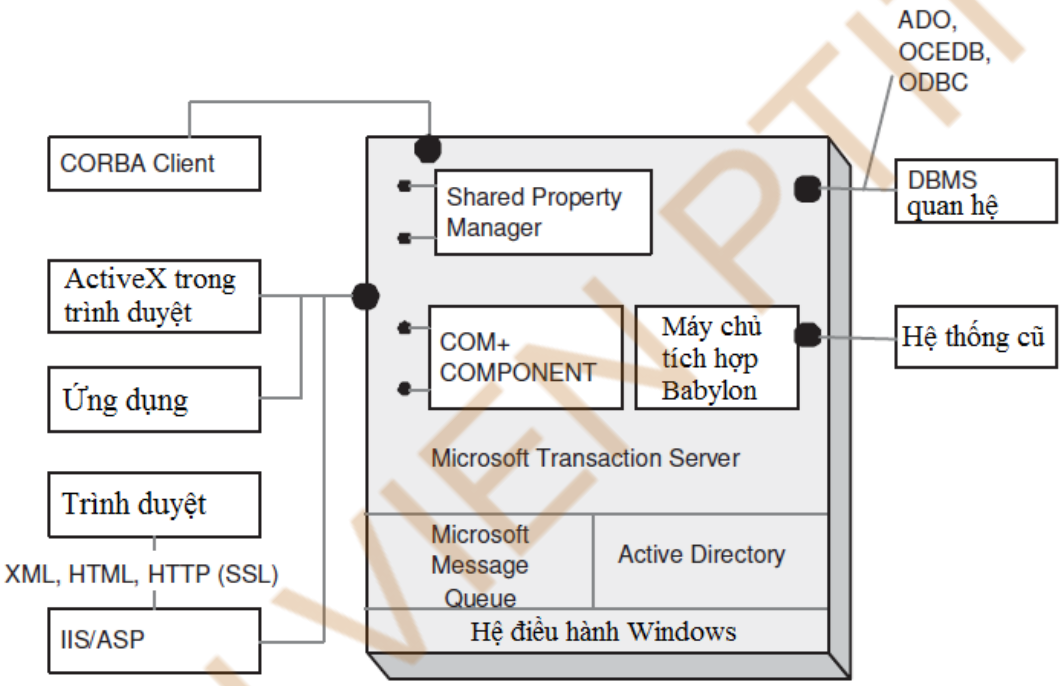
Hình 3.1: Kiến trúc J2EE

J2EE được minh họa trong hình 3.1. J2EE cung cấp các thành phần và chức năng sau đây:

- Phiên, Thực thể, Message Driven Beans (EJB);
- Quản lý giao dịch (JTA/JTS);
- Đặt tên và thư mục (JNDI);
- Remote Method Invocation (RMI);
- An ninh (JNDI Security); Java Authentication and Authorization Service (JAAS);
- Java Messaging Service (JMS);
- J2EE Connector Architecture (JCA);
- An ninh (realms, access control lists);
- Dịch vụ Web;
- XML (JAXP);
- Caching;
- Theo dõi và quản lý trên nền Web.

3.1.2 .NET

.NET là công nghệ nền tảng (framework) của Microsoft cho phép phát triển và tích hợp các ứng dụng cũng như các dịch vụ Web. Cấu trúc của nó được thể hiện trong hình 3.2. .NET Framework bao gồm bộ thông dịch (interpreter) và biên dịch (compiler) và một thành phần Common Language Runtime (CLR) để nhận các mã ứng dụng theo Microsoft Intermediate Language (MSIL). CLR chuyển đổi mã MSIL thành mã nguồn gốc (native) sử dụng kỹ thuật biên dịch just-in-time. Có nhiều bộ thông dịch có thể tạo ra MSIL cho nhiều ngôn ngữ lập trình, chẳng hạn như C++, Cobol và Visual Basic. Kết quả là các ứng dụng được viết bằng nhiều ngôn ngữ khác có thể được biên dịch thành MSIL và sau đó trở thành một phần của một dịch vụ Web.



Hình 3.2: Kiến trúc Microsoft .NET

Các máy chủ giao dịch Microsoft (Microsoft Transaction Server - MTS) tương ứng với các máy chủ J2EE. Chúng lưu trữ các ActiveX và đối tượng COM+, tương ứng với các EJB. .NET Active Service Directory (ADS) tương ứng với JNDI của J2EE, Microsoft Message Queue (MSMQ) tương ứng JMS J2EE, và chuẩn kết nối cơ sở dữ liệu mở (Open Database Connectivity hay ODBC) tương ứng JDBC. Cần lưu ý là các phương pháp tiếp cận .NET và J2EE là khá tương tự như nhau. Điểm khác biệt là .NET là một sản phẩm từ một nhà cung cấp duy nhất (Microsoft), trong khi J2EE là một đặc tả mà các sản phẩm có thể được tạo ra bởi các nhà cung cấp khác nhau.

3.2 TƯƠNG TÁC GIỮA CÁC THÀNH PHẦN DỊCH VỤ

Web Services Interoperability Organization (WS-I) là một tổ chức công nghiệp nhằm thúc đẩy khả năng tương tác ở mức trên các tiêu chuẩn thích hợp. Thành viên WS-I bao gồm một số các nhà cung cấp dịch vụ Web hàng đầu như IBM, Microsoft, BEA, và Sun. WS-I tạo ra các khuyến nghị về các tiêu chuẩn mà bản chất là đóng gói các tiêu chuẩn này thành một bộ tiêu chuẩn tương thích. Những khuyến nghị này được gọi là các cấu hình.

Hiện nay, các WS-I đã phát triển một hồ sơ được gọi là hồ sơ cơ bản 1.0. Hồ sơ này bao gồm SOAP 1.1, WSDL 1.1, 1.0 XML, XML Schema, và HTTP 1.1. Hơn nữa, hồ sơ cơ bản WS-I 1.0 quy định các giới hạn sau:

- SOAP chỉ nên được sử dụng với ràng buộc HTTP POST của nó.
- Các tiêu đề SOAPAction trong HTTP POST nên có một chuỗi trích dẫn.
- Bên nhận SOAP nên trả về một đáp ứng HTTP ngay sau khi nhận được một yêu cầu HTTP, và đáp ứng này nên là một mã HTTP thành công (200 hoặc 202) hoặc một mã lỗi HTTP. Các đáp ứng HTTP không được chứa một SOAP envelope.
- Bên yêu cầu SOAP phải bỏ qua bất cứ SOAP envelope mà có thể được trả lại bởi bên nhận SOAP.
- Các mẫu thông điệp WSDL chỉ được giới hạn là request-response và one-way.
- Chỉ mã hóa XML Schema được công nhận, không công nhận SOAP Section 5.

Lập trình viên không cần phải làm việc với các giao thức trực tiếp; thay vào đó, họ nên thực hiện các dịch vụ Web thông qua bộ công cụ phù hợp và giao diện lập trình. Do đó, các tiêu chuẩn như WS-I Basic Profile 1.0 là một mối quan tâm trực tiếp lớn nhất đối với các nhà phát triển công cụ.

3.3 PHÁT TRIỂN VÀ SỬ DỤNG DỊCH VỤ WEB

Phần này giới thiệu về phương pháp phát triển và sử dụng dịch vụ web dựa trên nền tảng .NET.

Như ta đã biết, một dịch vụ Web có thể được truy cập bằng các giao thức của web và được sử dụng bởi các ứng dụng web. Có ba bước trong phát triển Web

- Tạo dịch vụ web (creating)
- Tạo proxy
- Sử dụng dịch vụ web (consuming)

3.3.1 Tạo dịch vụ Web

Một dịch vụ Web là một ứng dụng Web về cơ bản giống như một lớp (class) mà bao gồm các phương thức (methods) có thể sử dụng bởi các ứng dụng khác. Về mặt lập trình, nó khá giống với các trang Web như các trang dựa trên nền tảng ASP.NET, nhưng dịch vụ web không có giao diện người dùng.

Để hiểu rõ khái niệm này; ta lấy ví dụ về tạo một dịch vụ web đơn giản có tên là StockService [5]. Nó cung cấp thông tin về giá cả trên thị trường chứng khoán. Người dùng có thể truy vấn về tên và giá chứng khoán bằng cách sử dụng các mã chứng khoán của các công ty đã niêm yết. Ví dụ mã chứng khoán của công ty cổ phần đầu tư và phát triển công nghệ là FPT. Để đơn giản, các giá trị được gán bằng tay (hardcoded) trên một mảng hai chiều. Dịch vụ web này sẽ có 3 phương thức:

- HelloWorld
- GetName

- GetPrice

Trước hết, ta thực hiện các bước sau để tạo một dịch vụ web mới

Bước 1: chọn File->New->Website trong bộ công cụ Visual Studio và chọn ASP.NET Web Service

Bước 2: Sẽ xuất hiện một tên dịch vụ (ngầm định) là Service.asmx ta đổi tên thành StockService.asmx; đồng thời đổi tên Service.cs thành StockService.cs trong thư mục App_Code. Sau đó hướng chỉ thị sau vào file .asmx.

```
<%@ WebService Language="C#"
                  CodeBehind="~/App_Code/StockService.cs"
                  Class="StockService" %>
```

Bước 3: Mở file StockService.cs để bổ sung thêm C# code vì nội dung code của file này được tự động sinh ra ngầm định (từ phương thức HelloWorld). Ta thêm vào một mảng hai chiều gồm các xâu kí tự là các mã chứng khoán và giá của chúng. Đồng thời ta cần viết mã cho hai phương thức GetName và GetPrice để lấy ra mã chứng khoán và giá của chúng.

```
using System;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// [System.Web.Script.Services.ScriptService]
public class StockService : System.Web.Services.WebService
{
   public StockService () {
     //Uncomment the following if using designed components
     //InitializeComponent();
   }
   string[,] stocks =
   {
   {"FPT", "Công ty Cổ phần FPT", "60,450"},
   {"KTB", "Công ty Khoáng sản Tây Bắc", "52,556"},
   {"MSN", "Công ty Công ty Masan", "120,225"},
   {"NTP", "Công ty Nhựa tiền phong", "45,365"},
   {"TTC", "Công ty Gỗ trường thành", "11,110"}
   };

   [WebMethod]
   public string HelloWorld() {
      return "Hello World";
   }
   [WebMethod]
   public double GetPrice(string symbol)
   {
      //it takes the symbol as parameter and returns price
      for (int i = 0; i < stocks.GetLength(0); i++)
      {
         if (String.Compare(symbol, stocks[i, 0], true) == 0)
         return Convert.ToDouble(stocks[i, 2]);
      }
      return 0;
   }
   [WebMethod]
   public string GetName(string symbol)
   {
      // It takes the symbol as parameter and
      // returns name of the stock
      for (int i = 0; i < stocks.GetLength(0); i++)
      {
         if (String.Compare(symbol, stocks[i, 0], true) == 0)
           return stocks[i, 1];
      }
      return "Không tìm thấy Mã chứng khoán: " + symbol;
   }
}
```

Hình 3.3: Mã chương trình của dịnh vụ Web StockService

Bước 4: Chạy thử dịch vụ

Khi chạy thử dịch vụ bằng cách gõ liên kết đến tên của dịch vụ web; chẳng hạn http://localhost:1080/WebService/StockService.asmx từ cửa sổ URL của trình duyệt ta thấy xuất hiện như hình 3.4

Hình 3.4: Chạy thử dịnh vụ Web StockService [5]

Khi chọn các phương thức GetName (hoặc GetPrice) rồi gõ giá trị vào kết quả sẽ hiển thị đúng tên công ty khi ta gõ đúng tên mã chứng khoán.

3.3.2 Sử dụng dịch vụ Web

Để sử dụng được dịch vụ Web vừa tạo. Ta cần tạo một trang web để gọi đến (invoke) các phương thức của dịch vụ Web. Giả sử ta có thể tạo một trang có giao diện đơn giản gồm một nhãn (label control) để hiển thị các kết quả trả về hai nút bấm (button control) để quay lại và để gọi hàm của dịch vụ. Mã chương trình của trang web này được trình bầy ở hình 3.5

```
<%@ Page Language="C#"
         AutoEventWireup="true"
         CodeBehind="Default.aspx.cs"
         Inherits="wsclient._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
    <div>
    <h3>Using the Stock Service</h3>
    <br />
    <br />
    <asp:Label ID="lblmessage" runat="server"></asp:Label>
    <br />
    <br />
    <asp:Button ID="btnpostback" runat="server"
                onclick="Button1_Click"
                Text="Post Back" style="width:132px" />

    <asp:Button ID="btnservice" runat="server"
                onclick="btnservice_Click"
                Text="Get Stock" style="width:99px" />
    </div>
</form>
</body>
</html>
```

Hình 3.5: Mã chương trình cho giao diện trang web [5]

Tạo proxy

Trước khi sử dụng được dịch vụ thì proxy cần được tạo ra để đăng kí với ứng dụng khách (client). Proxy cho phép gọi đến dịch vụ Web bằng các phương thức cục bộ. Proxy nhận lời gọi hàm và đổi chúng thành yêu cầu SOAP để gửi đến máy chủ web. Máy chủ web trả về các gói SOAP đến client và proxy sẽ giải mã các gói (package) SOAP và hiển thị chúng trên ứng dụng client.

Mã proxy cho sử dụng dịch vụ Web StockService này được trình bày trong hình 3.6.

```
protected void btnservice_Click(object sender, EventArgs e)
{
  StockService proxy = new StockService();
  lblmessage.Text = String.Format("FPT có giá: {0}",
  proxy.GetPrice("FPT").ToString());
}
```

Hình 3.6: Mã proxy [5]

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

//this is the proxy
using localhost;

namespace wsclient
{
   public partial class _Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
     {
        if (!IsPostBack)
           lblmessage.Text = "First Loading Time: " +
           DateTime.Now.ToLongTimeString();
        else
           lblmessage.Text = "PostBack at: " +
           DateTime.Now.ToLongTimeString();
     }
     protected void btnservice_Click(object sender, EventArgs e)
     {
       StockService proxy = new StockService();
       lblmessage.Text = String.Format("FPT có giá:{0}",
       proxy.GetPrice("FPT").ToString());
     }
   }
}
```

Hình 3.7: Mã chương trình goi đến (invoke) các hàm của dịch vụ Web [5]

3.4 CÔNG CỤ CHO PHÁT TRIỂN DỊCH VỤ WEB

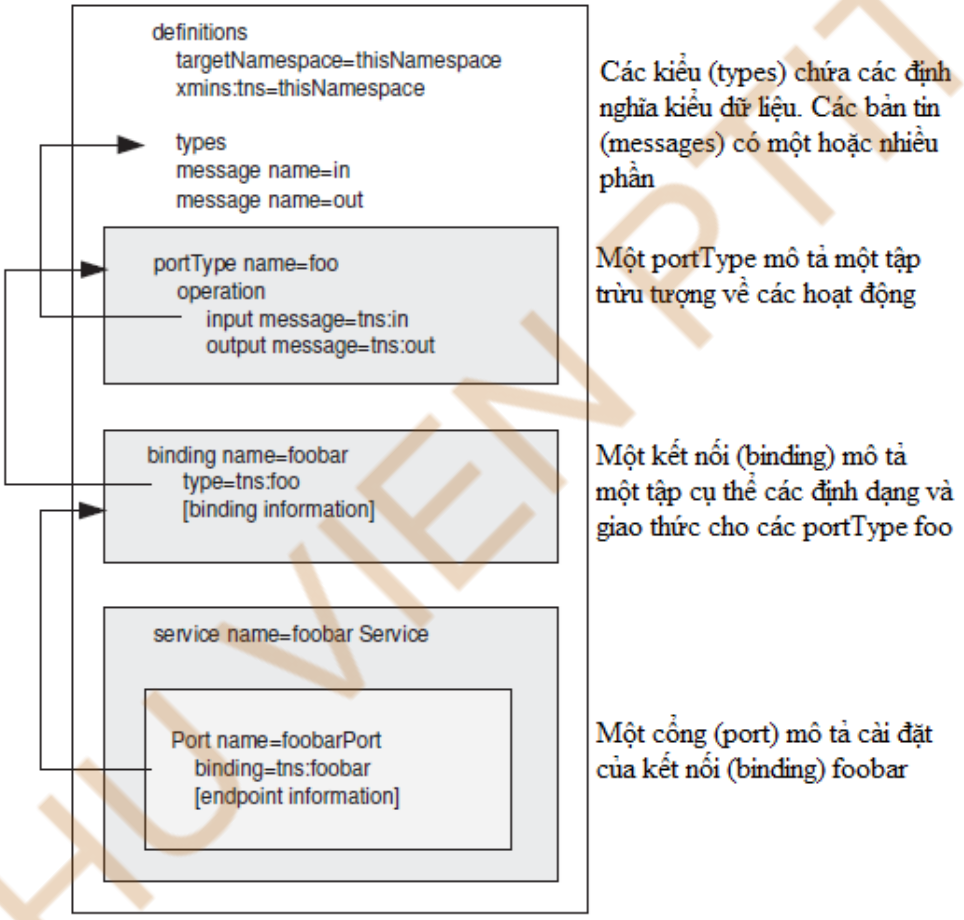
Về phía máy chủ, các lập trình viên có thể thực hiện lập trình cho logic kinh doanh và tạo ra các mô tả dịch vụ từ các logic kinh doanh đó. Các đặc tả WSDL có thể được tạo ra dễ dàng từ các ngôn ngữ phổ biến như Java, C# và các ngôn ngữ khác. Các dịch vụ này có thể được truy cập qua SOAP. Đồng thời, các đặc tả WSDL có thể sẵn sàng cho các máy khách sau này.

Về phía máy khách, các lập trình viên sẽ lấy những mô tả dịch vụ và áp dụng các công cụ chung để ánh xạ WSDL vào giao diện của các ngôn ngữ lập trình mong muốn của họ. Sau đó, họ sẽ tạo ra ứng dụng bằng cách sử dụng các giao diện này và cuối cùng, họ chạy các ứng dụng sử dụng SOAP để gọi các dịch vụ được cung cấp bởi máy chủ.

Để kích hoạt kết nối động, máy chủ sẽ sử dụng một giao diện lập trình công bố các dịch vụ để máy khách có thể sử dụng một giao diện lập trình tìm ra các dịch vụ cần thiết.

3.4.1 Lập trình WSDL

WSDL trông có vẻ phức tạp, nhưng thực ra là một khái niệm đơn giản. Hình 3.8 mô tả một cách đơn giản về tổng quan các thành phần trong một tập tin WSDL. Nó được thiết kế không phải chỉ để có thể đọc hay tóm tắt mà để dùng cho máy tính xử lý. Kết quả là, nó đơn giản đối với các công cụ để tạo ra WSDL tự động từ mã nguồn, đặc biệt nếu là hướng đối tượng. Các công cụ như Visual Studio .NET và Oracle Developer của Microsoft cung cấp chức năng này. Khi lập trình thực hiện hoặc sửa đổi mã nguồn của họ, chúng giúp tự động tạo ra các đặc tả WSDL.



Hình 3.8: Một khung nhìn đơn giản của mô hình dữ liệu WSDL

Tuy nhiên, có một điểm cần thận trọng là các công cụ tiết lộ các chi tiết thực hiện của các framework hướng đối tượng. Đặc biệt, việc lộ các đối tượng kinh doanh ra bên ngoài rất nguy hiểm, bởi vì những đối tượng nói chung được thiết kế và hoạt động hạn chế cho mục đích nội bộ. Việc phơi bày giao diện ra bên ngoài có nghĩa là hành vi của chúng có thể không được đầy đủ và có thể bị can thiệp vào hoạt động nội bộ theo một cách nào đó.

3.4.2 Lập trình Web Service với Java

Có một vài công cụ cho các dịch vụ Web như công cụ mã nguồn mở Apache eXtensible Interaction System (Axis) là SOAP engine, trong đó cũng bao gồm các chức năng quan trọng cho WSDL.

JAX-RPC và SAAJ

Java API cho RPC XML-Based (JAX-RPC) và SOAP với Attachments API cho Java (SAAJ) cung cấp giao diện lập trình ứng dụng Java để xử lý các thông điệp SOAP. JAX-RPC cao cấp hơn và được xây dựng trên SAAJ. JAX-RPC xử lý chuyển đổi giữa các đối tượng Java và XML, thực hiện kiểm tra kiểu trong chuyển đổi. JAX-RPC cũng bao gồm các công cụ để tạo ra các tài liệu WSDL từ mã Java và từ các tài liệu WSDL.

Đối với SOAP, SAAJ tự xây dựng trên JAXP và cung cấp một API đơn giản hướng về SOAP. Ví dụ, các API bao gồm các phương pháp để quản lý các kết nối SOAP, tạo thông điệp SOAP, trích xuất nội dung thích hợp (tiêu đề, phần thân), và xử lý các phản hồi.

Java API cho việc gửi thông điệp XML (JAXM) cung cấp các API cho việc tạo và xử lý bản tin SOAP. JAXM ở mức thấp hơn so với JAX-RPC và đã được thay thế bởi SAAJ.

Web Services Invocation Framework

Dịch vụ Web Services Invocation Framework của dự án Apache (WSIF) là một tiếp cận để gọi dịch vụ dựa trên WSDL. WSIF mang quan điểm là client. Tuy nhiên, nó được dựa trên mô tả WSDL của các dịch vụ và về nguyên tắc, là độc lập với các kết nối (binding). Một kết nối cho SOAP có sẵn và quan trọng, nhưng một WSIF client có thể dễ dạng chuyển sang một ràng buộc kết nối khác.

JAXR

Java API XML cho XML Registries (JAXR) là một cách tiếp cận dựa trên Java để truy cập vào nhiều loại registry khác nhau, bao gồm cả ISO 11.179, OASIS, ebXML, và UDDI. Nó hữu ích nhất cho việc truy cập một UDDI hoặc ebXML registry để quảng cáo hoặc tìm kiếm một dịch vụ.

JAXP

JAXP là một API để xử lý các tài liệu XML. Một trong những thành phần của JAXP là một công cụ phân tích cú pháp dựa trên DOM (Document Object Model). DOM cung cấp một khái niệm đơn giản để duyệt cây phân tích cú pháp của một tài liệu qua xử lý đệ quy. Có một công cụ phân tích cú pháp khác nữa là dựa trên Simple API cho XML (SAX). Hơn nữa, JAXP hỗ trợ các biến đổi XSL (XSLT).

3.5 BÀI TẬP

1. Phát triển một dịch vụ Web sử dụng công nghệ nền tảng .NET để cung cấp dịch vụ tra cứu điểm các môn học trong một học kỳ cho sinh viên đại học. Bạn được yêu cầu tạo một cơ sở dữ liệu sinh viên trên hệ quản trị cơ sở dữ liệu do bạn lựa chọn (như SQL server, MS Access, MySQL v.v..). Cơ sở dữ liệu này có các bảng chứa các thông tin về sinh viên như: mã sinh viên, họ và tên sinh viên, học kỳ (học kỳ mấy?), danh sách các môn học theo từng học kỳ và điểm trung bình của từng môn học.

   Trong dịch vụ Web có một phương thức là Tracuu với 2 tham số đầu vào là mã sinh viên và học kỳ; phương thức này sẽ trả về tên sinh viên, danh sách các môn học và điểm của từng môn theo học kỳ.

Học viên cũng được yêu cầu phát triển ứng dụng phía client để gọi đến (invoke) phương thức Tracuu và hiển thị thông tin trả về dưới dạng một bảng.

2. Thực hiện như bài tập số 1; nhưng công cụ sử dụng là nền tảng Java

CHƯƠNG 4
CÁC NGUYÊN LÝ TÍNH TOÁN HƯỚNG DỊCH VỤ

4.1 CÁC THỂ HIỆN ỨNG DỤNG CỦA DỊCH VỤ WEB

Điện toán hướng dịch vụ (service-oriented computing) không phải là một tập hợp các công nghệ của tương lai cho các ứng dụng, mà là một tập các công nghệ hiện có hoặc mới để giải quyết vấn đề còn tồn tại trong khoa học máy tính.

Đầu tiên, chúng ta xem xét tại một số thời điểm công tác của một bộ phận phẫu thuật điển hình trong một bệnh viện lớn. Thách thức ở đây sẽ là lập hóa đơn, lập lịch và thanh toán. Mỗi một hệ thống có khả năng sẽ khá phức tạp và liên quan đến giao diện người dùng và cơ sở dữ liệu riêng của mình, chạy trên các hệ điều hành khác nhau.

Có những lý do rõ ràng để thấy rằng những hệ thống này cần tương tác với nhau. Ví dụ, lập kế hoạch cho nhân sự và điều hành cho phòng phẫu thuật là một công việc phức tạp. Lịch hiếm khi được hoàn chỉnh mà thường xuyên được cập nhật.

Tiếp theo, các nhân viên phải được trả lương cho những nỗ lực của họ thông qua hệ thống tính lương. Cũng như đối với lập kế hoạch, cơ chế trả lương rất phức tạp, bởi vì các loại quy tắc làm thêm giờ phải được xem xét cho các loại lao động khác nhau: y tá, nghiên cứu sinh, các bác sĩ tư vấn, bác sĩ phẫu thuật cấp cao, gây mê, bác sĩ X quang, v.v.

Tương tự như vậy, các hệ thống thanh toán cũng phải kết hợp với các thông tin lịch trình. Việc thanh toán cực kỳ phức tạp trong hầu hết các doanh nghiệp, đặc biệt là thanh toán thuốc. Hệ thống thanh toán không chỉ là hóa đơn cho khách hàng, mà còn dùng để làm việc với các công ty bảo hiểm y tế và các cơ quan chính phủ (ví dụ, cơ quan cho trẻ em, người già, nhân viên nhà nước về hưu, cựu chiến binh). Chú ý rằng việc thanh toán cho một ca phẫu thuật của bệnh nhân phụ thuộc không chỉ vào các phẫu thuật mà còn vào các sự kiện dường như không liên quan khác. Ngược lại, hệ thống lập lịch có thể làm việc với một số công cụ hỗ trợ quyết định khác để đảm bảo rằng các hóa đơn tổng thể của bệnh viện được tối ưu hóa.

4.1.1 Việc phối hợp các ứng dụng trong một doanh nghiệp

Trong một doanh nghiệp có thể sử dụng các chương trình ứng dụng khác nhau. Vấn đề ở đây là làm thế nào để các thành phần phần mềm khác nhau có thể làm việc tốt với nhau, trong khi các thành phần này thông thường hoạt động như hệ thống độc lập khép kín.

Ở đây, chỉ quan tâm đến khía cạnh phối hợp của vấn đề nêu trên và có thể dễ dàng nhận thấy rằng có một số khó khăn cần phải vượt qua. Đầu tiên, cần phải có khả năng kết nối giữa các ứng dụng. Điều này được đảm bảo nhờ sự phổ biến của mạng IP cùng với các giao thức cao cấp hơn như TCP và HTTP. Thứ hai, phải có thứ gì đó cho phép các thành phần khác nhau có thể hiểu cách thức trao đổi thông tin của nhau. Hiện nay, nhờ vào định dạng cách thức trao đổi thông qua việc sử dụng XML, đặc biệt là với sự trợ giúp của XML Schemas về quy định cú pháp, vấn đề này đã được giải quyết. Tuy nhiên, vẫn còn những khó khăn khác, ví dụ như ý nghĩa của các thông tin trong liên lạc. Ý nghĩa thường không được đưa ra một cách rõ ràng, mà phụ thuộc vào cách các hệ thống xử lý thông tin. Do đó, một trong những khó khăn

lớn nhất cần phải giải quyết được trong việc phối hợp ở đây là việc phân giải ngữ nghĩa trong giao dịch (tương tác) giữa các thành phần.

Tính toán hướng dịch vụ cung cấp các công cụ để mô hình hóa thông tin và liên kết các mô hình, xây dựng các quy trình trong hệ thống, xác nhận và bảo đảm các thuộc tính giao dịch, bổ sung các hỗ trợ quyết định linh hoạt, và liên kết hoạt động của các hệ thống phần mềm thành phần với các tổ chức mà chúng đại diện.

4.1.2 Việc phối hợp các ứng dụng giữa các doanh nghiệp

Việc phối hợp các ứng dụng giữa nhiều doanh nghiệp đang nhanh chóng trở thành một vấn đề quan trọng trong ứng dụng Công nghệ thông tin. Trước đây, việc phối hợp các ứng dụng giữa các doanh nghiệp thường phải dùng các phương pháp cơ học, cần sự can thiệp phần lớn của con người; hoặc là phải sử dụng các tiêu chuẩn cứng nhắc như Electronic Data Interchange (EDI), dẫn đến khó khăn trong việc bảo trì hệ thống. Trong những năm gần đây, ngày càng có nhiều mối quan tâm trong việc quản lý chuỗi cung ứng và sản xuất thông minh, do đó cần đến quá trình phối hợp doanh nghiệp, nghĩa là các doanh nghiệp phải làm việc cùng nhau. Do vậy, nếu có thể sắp xếp các giao dịch thông qua công nghệ, các doanh nghiệp có thể cải thiện việc đáp ứng với thông tin, giảm chi phí và khai thác được nhiều cơ hội mới.

Hãy xem xét kịch bản y tế đã nêu trên một lần nữa. Như đã lưu ý ở trên, một bệnh viện cần làm hóa đơn cho các thực thể bên ngoài. Đây có thể là các công ty bảo hiểm và các cơ quan chính phủ (ngoại trừ bệnh nhân ra). Phương pháp truyền thống là gửi hóa đơn giấy in, và sẽ được nhập lại bởi bên nhận. Tuy nhiên, cách tiếp cận này phần lớn sẽ biến mất nhờ việc ứng dụng thanh toán trực tuyến.

Nhưng xử lý trực tuyến tạo ra một số khó khăn trong việc phối hợp ứng dụng. Các định dạng dữ liệu cần phải được lưu trữ theo cách tin cậy, khi đó thông tin định dạng của bệnh viện có thể được hiểu bởi các công ty bảo hiểm và ngược lại. Tuy nhiên, những tính toán về năng suất chỉ ra rằng nên sử dụng một cách tiếp cận tiêu chuẩn, để từ đó các định dạng có thể được xử lý thông qua các công cụ thương mại mạnh mẽ, chứ không phải thông qua phần mềm tùy chỉnh. Trong những năm gần đây, ngành công nghiệp đã hội tụ về XML là định dạng dữ liệu lựa chọn. Đây rõ ràng là một thành công. Tuy nhiên, nó mở ra những câu hỏi rằng làm cách nào để nội dung của các dữ liệu truyền có thể được hiểu và xử lý.

Tính toán hướng dịch vụ cung cấp các lợi ích tương tự như đối với phối hợp ứng dụng trong doanh nghiệp. Ngoài ra, nó cung cấp cho các bên giao dịch khả năng tự xây dựng hành vi để từ đó từng có thể tự chủ áp dụng chính sách nội bộ và đạt được các tiến trình phối hợp giữa các doanh nghiệp hiệu quả và chặt chẽ.

4.1.3 Cấu hình ứng dụng

Giả sử bệnh viện mua một hệ thống quản lý thông tin gây mê (AIMS) để bổ sung cho hệ thống hiện có. Một AIMS sẽ cho phép bác sĩ gây mê quản lý các thủ tục gây mê cho bệnh nhân phẫu thuật tốt hơn, nhằm theo dõi, ghi lại và báo cáo các hoạt động, như bật hoặc tắt các loại khí khác nhau hay nhỏ giọt. Những thông tin này có thể giúp thiết lập việc tuân thủ theo các quy định của chính phủ, đảm bảo rằng các hướng dẫn lâm sàng nhất định được đáp ứng, và hỗ trợ các nghiên cứu về kết quả của bệnh nhân. Việc mua AIMS là khá dễ dàng, nhưng

việc cài đặt và sử dụng lại không hề đơn giản. Để đưa ra sử dụng một hệ thống loại này đòi hỏi cả các hệ thống mới và hệ thống hiện có phải chọn ra được giao diện hợp lệ (right interface). Bởi vì các hệ thống này có thể đã được phát triển trên các nền tảng khác nhau và có thể, trên thực tế, chạy trên các hệ điều hành khác nhau, do đó việc tuân thủ các tiêu chuẩn cho các mối liên kết là cần thiết cho mỗi hệ thống.

Chúng ta có thể giả định rằng việc kết nối ở mức thấp đã được thực hiện. Tuy nhiên, vẫn còn những khó khăn khác giống như trong việc phối hợp các ứng dụng trong doanh nghiệp. Một là việc truyền thông điệp (messaging) để hệ thống có thể được kết nối khi hoạt động. Hai là cần giải quyết cấn đề về ngữ nghĩa để các thành phần hiểu nhau.

Tuy nhiên, có một thách thức khác khi đưa ra một ứng dụng mới. Đó là việc cấu hình và tùy chỉnh hành vi. Trong trường hợp của một AIMS, nó phải được gắn với một mô hình dữ liệu bệnh viện cụ thể hoặc các thuật ngữ để nó hiển thị màn hình giao diện người dùng chính xác cho các nhân viên bệnh viện và ghi lại những quan sát đúng. Mô hình này được điều chỉnh bởi các thủ tục của bệnh viện cũng như các yêu cầu áp đặt bởi các công ty bảo hiểm và các cơ quan chính phủ đang giao tiếp. Nếu ứng dụng được thiết kế với những cân nhắc sử dụng tính toán hướng dịch vụ, thì sau đó nó có thể được cấu hình một cách nhanh chóng và đưa vào quy trình kinh doanh hiện tại.

Tính toán hướng dịch vụ cho phép tuỳ biến các ứng dụng mới bằng cách cung cấp một giao diện dịch vụ Web giúp loại bỏ các vấn đề gửi thông điệp (messaging) đồng thời cung cấp một cơ sở ngữ nghĩa để tùy chỉnh các chức năng của ứng dụng.

4.1.4 Lựa chọn động

Giả sử bệnh viện muốn mua vật tư là ống thông. Để thực hiện việc mua sắm có hiệu quả bệnh viện cần phải có khả năng giao dịch (phối hợp ứng dụng) với các nhà cung cấp ống thông – đây là một trường hợp phối hợp các ứng dụng giữa các doanh nghiệp. Hiện tại, giả sử là bệnh viện có thể mua ống thông từ bất cứ nhà cung cấp nào có điều khoản tốt nhất, nghĩa là, có thể dễ dàng lựa chọn đối tác. Lựa chọn động kiểu này ngày càng trở nên phổ biến bởi sự linh hoạt của nó là rất hữu ích. Nếu các đối tác kinh doanh có thể được lựa chọn một cách linh hoạt, thì sau đó họ có thể được lựa chọn để tối ưu hóa bất kỳ loại tiêu chuẩn chất lượng dịch vụ nào, như hiệu suất, độ tin cậy và sự tin tưởng.

Tính toán hướng dịch vụ cho phép lựa chọn động các đối tác kinh doanh dựa trên tiêu chuẩn chất lượng dịch vụ mà mỗi bên có thể tùy chỉnh cho chính họ.

4.1.5 Khả năng chịu lỗi của phần mềm

Giả sử một bệnh viện đang thực hiện một giao dịch kinh doanh với một đối tác và gặp sự cố. Sẽ là rất tốt nếu các tương tác trong giao dịch có thể được sửa đổi cấu trúc động theo đối tác kinh doanh thay thế (mới) theo một cách trong suốt đối với toàn bộ quá trình. Mở rộng hơn nữa là khi trạng thái của tương tác bị mất, thì cần một vài phương pháp phục hồi để khôi phục lại trạng thái thích hợp và tiếp tục làm việc với các đối tác kinh doanh mới.

Tính toán hướng dịch vụ cung cấp hỗ trợ cho lựa chọn động các đối tác cũng như các khái niệm trừu tượng mà qua đó các trạng thái của một giao dịch kinh doanh có thể được lưu

giữ và xử lý linh hoạt; theo cách này, lựa chọn động được khai thác để mang lại khả năng chịu lỗi mức ứng dụng.

4.1.6 Lưới

Điện toán lưới (grid computing) dùng để tính toán phân tán với một số tài nguyên có sẵn trên mạng, và được kết hợp vào các ứng dụng lớn theo yêu cầu. Điện toán lưới là một hình thức của siêu tính toán và được phát triển như là sự kế thừa của phương pháp trước đây cho quy mô tính toán khoa học lớn. Việc xây dựng các ứng dụng phức tạp trên kiến trúc lưới là rất khó khăn, do vậy cần quan tâm đến việc mô đun hóa hơn nữa các giao diện dựa trên dịch vụ. Như vậy, các dịch vụ điện toán lưới đã được đề xuất tương đồng với các dịch vụ Web.

Tính toán hướng dịch vụ cho phép sử dụng hiệu quả các nguồn tài nguyên điện toán lưới.

4.1.7 Phát triển phần mềm

Phát triển phần mềm vẫn là một nỗ lực trí tuệ đầy thử thách. Những cải tiến được thực hiện thông qua việc sử dụng các khái niệm trừu tượng cao. Các dịch vụ cung cấp các trừu tượng lập trình, ở đó các mô đun phần mềm khác nhau có thể được phát triển thông qua các giao diện tốt hơn so với trước đây. Khi toàn bộ các mô tả ngữ nghĩa được sử dụng, các mô-đun không chỉ dễ dàng tùy biến hơn, mà còn có khả năng sau:

Tính toán hướng dịch vụ cung cấp một mô hình tính toán linh hoạt và phong phú về mặt ngữ nghĩa, giúp đơn giản hoá phát triển phần mềm.

4.2 TIẾN TRÌNH NGHIỆP VỤ

4.2.1 Tiến trình nghiệp vụ (business process)

Một tiến trình là một hoạt động. Nói chung một tiến trình sẽ là một hoạt động tổng hợp và được hướng đến để phục vụ một số mục đích. Tùy thuộc vào các tiến trình cụ thể mà nhiệm vụ của chúng có thể là sự kết hợp của các dịch vụ, tương ứng với các truy vấn, giao dịch, các ứng dụng, và các hoạt động hành chính. Những dịch vụ này có thể phân tán trong hoặc giữa các doanh nghiệp và sẽ được điều phối bằng việc ràng buộc các điều khiển và dữ liệu giữa chúng. Các dịch vụ có thể tự chúng là phức hợp, ví dụ như được cài đặt như các tiến trình. Các thảo luận theo sau nhấn mạnh rằng quy trình kinh doanh bao gồm các dịch vụ, nhưng những khái niệm phát triển có thể áp dụng như nhau cho tính toán khoa học và các thiết lập khác. Ví dụ về các thiết lập mà ở đó ta áp dụng tiến trình bao gồm các môi trường trong doanh nghiệp (tức là, trong một doanh nghiệp), như lập kế hoạch sản xuất và kiểm soát hàng tồn kho, và môi trường liên doanh nghiệp (tức là, giữa các doanh nghiệp), chẳng hạn như quản lý chuỗi cung ứng và đàm phán mua. Rõ ràng là các tiến trình trong doanh nghiệp và liên doanh nghiệp cần phải tương quan với nhau, bởi vì hoạt động trong doanh nghiệp là cần thiết để hỗ trợ tương tác liên doanh nghiệp.

Tiến trình đưa ra một số thách thức về kỹ thuật. Đầu tiên, chúng ta phải có khả năng mô hình hóa một tiến trình, kết hợp sự chính xác trong việc thực thi với mô hình, và tôn trọng những hạn chế của các dịch vụ cơ bản và các nguồn lực của chúng. Các thực thi bình thường

của một quá trình thường dễ dàng, vì chúng có thể đơn giản như một trật tự một phần của các hoạt động trong tiến trình này. Ngược lại, các điều kiện ngoại lệ có thể là khó để mô hình và xử lý. Quan trọng hơn, vì các quá trình kinh doanh quan trọng thường chạy lâu, tương tác lẫn nhau là không đơn lẻ, dẫn đến khả năng thông tin đầu vào có thể được xem xét lại và do đó gây ra kết quả là vô hiệu. Các ngoại lệ và sửa đổi là những nguyên nhân chính của những khó khăn trong việc mô hình hóa một tiến trình.

Thứ hai, chúng ta phải có khả năng tạo giao diện cho các chức năng cơ bản của một tiến trình. Trong trường hợp của các hệ thống quản trị cơ sở dữ liệu (DBMS), nó sẽ bao gồm các mô hình phù hợp của các giao dịch, kết hợp các ràng buộc về kiểm soát đồng thời và cơ chế phục hồi DBMS. Một mô hình giao dịch cung cấp các trừu tượng cần thiết và bảo vệ các mô hình tiến trình khỏi những chi tiết trong cài đặt của DBMS.

Bởi vì các tiến trình được sử dụng trong một số nơi của doanh nghiệp để hỗ trợ hoạt động nội bộ cũng như tương tác với các đối tác kinh doanh, chúng có thể được mô hình hóa theo những cách khác nhau, thường dựa trên các biểu diễn tiến trình riêng của các nhà cung cấp phần mềm và các ứng dụng liên quan. Ví dụ, nếu phần mềm lập lịch sản xuất sử dụng một hình thức mô hình khác với phần mềm xử lý đơn đặt hàng, thì việc tham gia của doanh nghiệp trong một chuỗi cung ứng có thể bị ảnh hưởng xấu. Tuy nhiên, phối hợp giữa các quá trình, trong khi rõ ràng là một nhu cầu quan trọng trong bối cảnh thực tế, gần như là không thể không có một bộ phiên dịch giữa các mô hình tiến trình. Những thách thức của tính không đồng nhất đã thảo luận trong bối cảnh chia sẻ thông tin sẽ áp dụng giống như đối với tương tác mô hình tiến trình.

Trước khi đi vào chi tiết, ta cần mô tả các quan điểm chính có thể có về tiến trình và sự phân biệt giữa chúng.

- Orchestration - Xem một tiến trình như một chương trình hay một trật tự một phần của các hoạt động cần thực thi. Quan điểm này xem một tiến trình như một công cụ "hợp tấu". Đó là, nếu như các đặc tả tiến trình đang được thực hiện dưới sự kiểm soát hay thay mặt cho một bên cụ thể. Orchestration tương ứng nhất là các diễn tả luồng công việc và ngôn ngữ tiến trình như BPEL4WS.
- Choreography - Xem một tiến trình như là các trao đổi thông điệp giữa các thành viên tham gia. Việc trao đổi thông điệp bị hạn chế để diễn ra trong các chuỗi khác nhau và có thể được yêu cầu phải được nhóm lại thành các giao dịch khác nhau. Choreography tương ứng nhất với các ngôn ngữ như WSCL và WSCI.
- Hợp tác (Collaboration) – Xem tiến trình như một sự hợp tác giữa các đối tác kinh doanh. Các đối tác kinh doanh không chỉ gửi tin nhắn cho nhau, mà còn có thể tham gia vào các mối quan hệ kinh doanh như hợp đồng và nghĩa vụ. Họ tạo ra trao đổi tin nhắn linh hoạt tùy thuộc vào hoàn cảnh phát triển và chính sách nội bộ của họ, ví dụ như, để xử lý các ngoại lệ kinh doanh. Hợp tác đang nổi lên như một cách tiếp cận nghiêm túc để thực hiện các tiến trình kinh doanh có quy mô lớn.

4.2.2 Mô tả tiến trình nghiệp vụ

4.2.2.1 Mô tả với UML

UML cung cấp các cấu trúc đồ họa có thể được sử dụng để mô tả (1) hành động và các hoạt động, và (2) các ưu tiên thời gian và luồng điều khiển. Các cấu trúc điều khiển được phép bao gồm:

• Tuần tự (Sequence), là một chuyển đổi từ một hoạt động này sang hoạt đông tiếp theo trong một thời điểm.

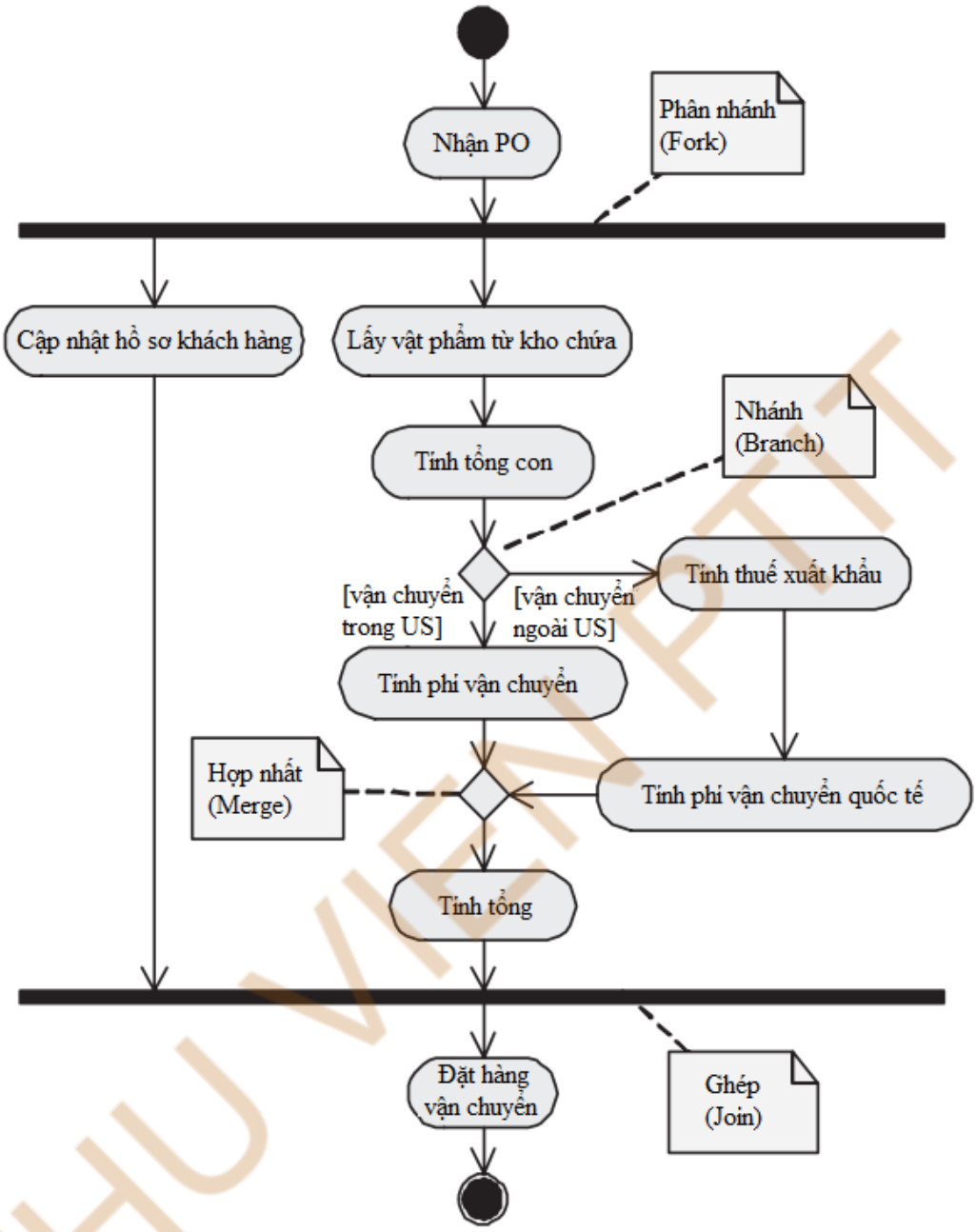
• Nhánh (Branch), là một điểm quyết định giữa các luồng điều khiển.

• Hợp nhất (Merge), nơi hai hay nhiều luồng điều khiển hợp lại.

• Phân nhánh (Fork), tách một luồng điều khiển thành hai hay nhiều luồng điều khiển đồng thời và độc lập.

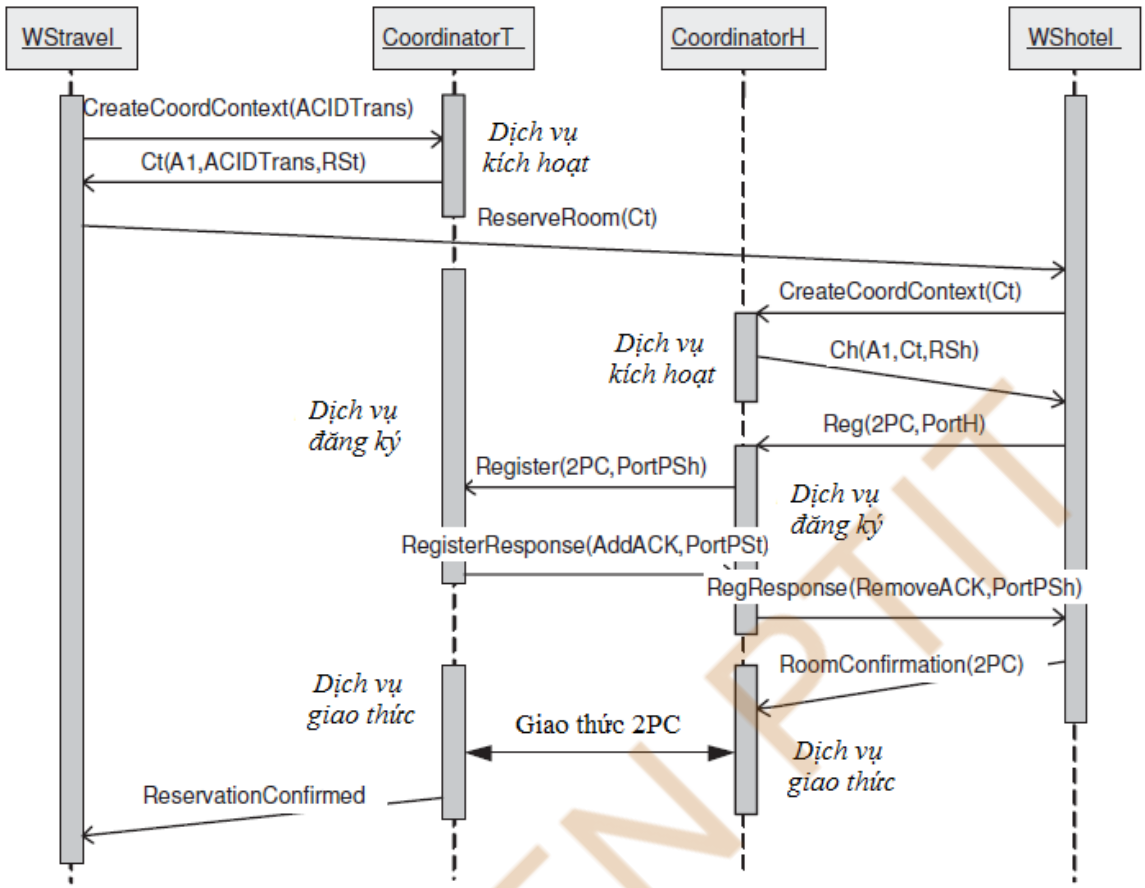
• Ghép (Join), đồng bộ hai hay nhiều luồng điều khiển thực hiện đồng thời thành một luồng.

Các cấu trúc điều khiển là một tập đầy đủ để mô tả một tiến trình hoặc luồng công việc tùy ý. Như vậy, chúng cũng có thể mô tả một dịch vụ Web tổng hợp. Một tiến trình cụ thể được mô tả bằng một biểu đồ hoạt động, một ví dụ trong số đó được thể hiện trong hình 4.1.



Hình 4.1: Một ví dụ về sơ đồ hoạt động UML

Một biểu đồ tuần tự UML được sử dụng để hiển thị các tương tác giữa các đối tượng đồng thời hiện hữu hoặc các tiểu trình đồng thời thực thi và các thể hiện tiến trình. Nó tập trung vào thứ tự thời gian của các thông điệp giữa các thực thể này. Hình 4.2 là một ví dụ về một biểu đồ tuần tự.



Hình 4.2: Sự phối hợp giữa hai dịch vụ Web

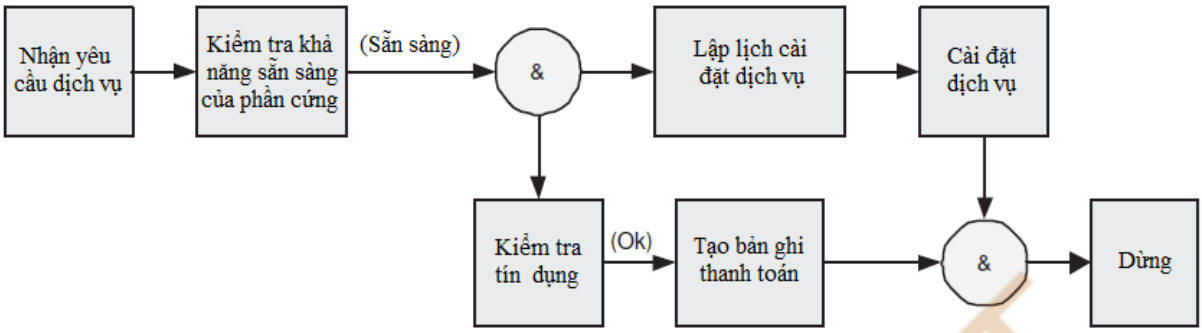
4.2.2.2 Luồng công việc

Một luồng công việc là một hoạt động nhằm giải quyết một số nhu cầu kinh doanh bằng cách thực hiện kiểm soát luồng dữ liệu cụ thể giữa các hoạt động con có liên quan đến tài nguyên thông tin và con người. Một ví dụ điển hình của một luồng công việc là xử lý nợ: khi bạn cần một khoản vay, bạn điền vào một mẫu đơn, một nhân viên bán hàng đánh giá đơn đăng ký, một kiểm toán viên xác minh các thông tin, và một giám sát viên gọi một cơ quan tín dụng bên ngoài hoặc sử dụng một công cụ đánh giá rủi ro tín dụng. Mỗi người trong quá trình cho vay nhận được thông tin liên quan đến yêu cầu của bạn, chỉnh sửa hoặc thêm vào, và chuyển tiếp các kết quả.

Trong tài liệu này, luồng công việc là một khái niệm hẹp hơn so với tiến trình. Tiến trình có thể được thực hiện thông qua luồng công việc, nhưng có thể thông qua các phương tiện khác như giao thức kinh doanh hoặc hội thoại giữa các agent. Định nghĩa trên về luồng công việc nhấn mạnh luồng dữ liệu và điều khiển giữa các hoạt động con là bản chất của công việc. Những luồng này cần thiết để nhìn ra các tiến trình mong muốn.

Cuối cùng, dù bạn chỉ định một tiến trình như thế nào đi nữa thì luồng dữ liệu và điều khiển sẽ xảy ra khi nó được ban hành. Điểm mấu chốt là trong công nghệ luồng công việc, luồng dữ liệu và điều khiển như vậy được xác định trực tiếp từ một quan điểm trung tâm một cách hợp lý. Giả định mô hình này có cả những điểm mạnh và điểm yếu của công nghệ luồng công việc. Các nhà cung cấp dịch vụ có thể quản lý các luồng công việc được sử dụng để cài đặt các dịch vụ nhất định. Một cài đặt dựa trên kỹ thuật luồng công việc có thể giúp quản lý

các ngoại lệ tiềm năng tốt hơn so với một ứng dụng truyền thống, nơi luôn ẩn những suy diễn cần thiết. Tuy nhiên, luồng công việc cũng có những hạn chế của chúng.



Hình 4.3: Một luồng công việc để xử lý trình tự dịch vụ viễn thông

Hình 4.3 minh họa ví dụ về một công việc được thực hiện khi bạn đặt hàng một dịch vụ từ một nhà cung cấp viễn thông. Bạn bắt đầu trình tự bằng cách tương tác với một đại diện bán hàng từ các nhà cung cấp, người đại diện giúp bạn điền vào một mẫu đơn. Đại diện bán hàng kiểm tra bằng một cơ sở dữ liệu để xác định xem liệu các phần cứng cần thiết đã có chưa. Nếu có, bạn nhận được một ước tính khi nào các dịch vụ này sẵn sàng để sử dụng. Bộ phận cài đặt dịch vụ địa phương sẽ đến cài đặt dịch vụ cho bạn, trong khi các nhà cung cấp viễn thông kiểm tra lịch sử tín dụng của bạn.

4.3 HỢP DỊCH VỤ

4.3.1 Hợp dịch vụ

4.3.1.1 Mục tiêu

Hầu hết các ứng dụng chào hàng cho các dịch vụ Web là các tương tác client-server đơn giản và dễ hiểu. Ví dụ, cơ sở dữ liệu chuyến bay lịch trình của một hãng hàng không có thể tương tác trực tiếp với các phần mềm lịch hẹn và thông tin cá nhân người dùng máy tính để đặt một chuyến bay; hoặc phần mềm cho một cơ sở dữ liệu địa chỉ liên lạc cá nhân có thể tự động truy vấn một loạt cơ sở dữ liệu điện thoại từ xa để thêm số điện thoại thiếu vào danh sách của mình. Những kiểu kịch bản như vậy không phải là quá xa vời. Ngay trong những ngày đầu của tiêu chuẩn dịch vụ Web, Southwest Airlines và Dollar Rent-A-Car đã phát triển một hệ thống nguyên mẫu có sử dụng SOAP để liên kết trang web Southwest đến hệ thống đặt phòng của Dollar, để khách hàng có thể đặt xe cùng với vé máy bay của họ.

Mặc dù hữu ích, các ứng dụng như vậy là không đủ để giúp phát triển và triển khai dịch vụ Web một cách mạnh mẽ. Các ứng dụng có hiệu quả, và cũng thách thức hơn, đòi hỏi các dịch vụ được kết hợp theo những cách sử dụng mới lạ hơn và với năng suất cao hơn. Ví dụ, lịch trình chuyến bay của hãng hàng không cũng có thể cho phép một nhà cung cấp nhiên liệu dự đoán việc mua nhiên liệu của hãng hàng không và cảnh báo nhà máy lọc dầu trong việc điều chỉnh tốc độ sản xuất. Hoặc, một trang web công ty du lịch có thể kết hợp các dịch vụ của Southwest Airlines, Dollar Rent-A-Car, và Sheraton để xây dựng các gói du lịch tùy chỉnh.

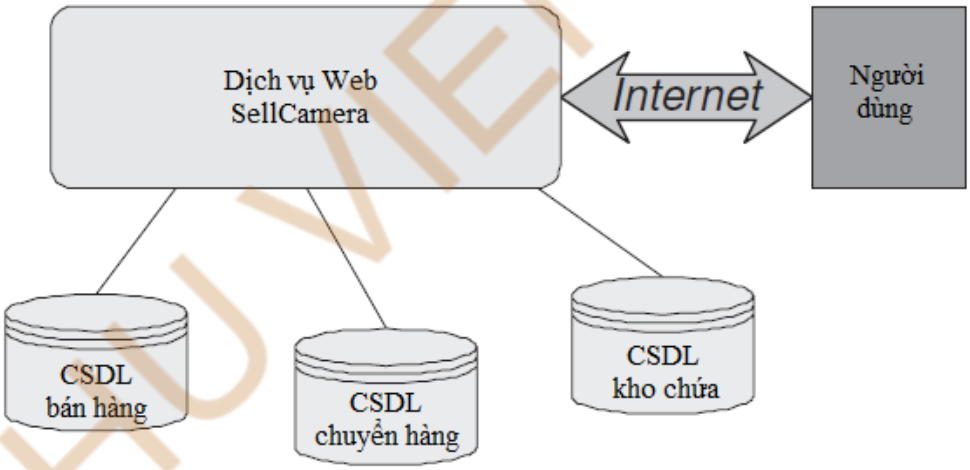
Hợp dịch vụ đã được nghiên cứu trong các tài liệu nghiên cứu trong một thời gian khá lâu, nhưng bây giờ nó đang trở thành một chủ đề quan trọng trong việc phát triển hệ thống

Web thực tế. Ý tưởng cơ bản đằng sau thành phần dịch vụ là đơn giản. Các trang web có thể được coi như là không chỉ cung cấp nội dung, mà còn cung cấp dịch vụ. Ví dụ, Yahoo! cung cấp một dịch vụ tin tức và Amazon cung cấp một dịch vụ lựa chọn sách. Chúng ta thường gọi các dịch vụ này bằng tay thông qua một trình duyệt Web, nhưng một chương trình có thể gọi trực tiếp.

Hợp dịch vụ trên Web đề cập đến việc lấy một số dịch vụ hiện có và xây dựng các dịch vụ tùy chỉnh mới. Ví dụ, bạn có thể tìm thấy những tiêu đề tin tức mới nhất và tìm kiếm những cuốn sách phù hợp với những tiêu đề. Hoặc, thông thường hơn, bạn có thể tạo ra một dịch vụ du lịch để gọi dịch vụ khách sạn, hãng hàng không, và cho thuê xe. Nói cách khác, bạn sẽ tạo ra một tiến trình làm việc với các dịch vụ hiện có.

4.3.1.2 Thách thức

Ưu điểm chính của dịch vụ Web biểu hiện là khi chúng ta có thể tận dụng chúng để tạo ra các dịch vụ mới. Thật không may, rất nhiều sự chú ý vào các dịch vụ Web đã được tập trung vào mức thấp, các vấn đề cơ sở hạ tầng, rồi đến cú pháp mã hóa và các công cụ cần thiết để gọi dịch vụ. Để dịch vụ Web có được hiệu quả đòi hỏi một sự hiểu biết về các khái niệm sâu sắc hơn. Các khái niệm này đã được phát triển ở các bộ phận khác nhau của khoa học máy tính, đặc biệt là cơ sở dữ liệu không đồng nhất, tính toán phân tán, trí tuệ nhân tạo, và các hệ thống đa tác tử (multiagent).



Hình 4.4: Ví dụ về môi trường giao dịch B2C

Hãy xem xét một trường hợp B2C đơn giản, nơi một công ty bán máy ảnh kỹ thuật số trên Web, kết hợp một danh mục online chứa các mẫu mới nhất cùng giá của chúng, một giao dịch thẻ tín dụng hợp lệ, và giao hàng bảo đảm. Phần mềm giao dịch B2C, như thể hiện trong hình 4.4, sẽ

- Ghi lại một giao dịch bán hàng trong cơ sở dữ liệu bán hàng.
- Ghi nợ thẻ tín dụng.
- Gửi đơn đặt hàng cho bộ phận vận chuyển.
- Nhận sự đồng ý từ các bộ phận vận chuyển cho việc giao hàng ngày hôm sau.
- Cập nhật cơ sở dữ liệu kho hàng.

Tuy nhiên, một số vấn đề có thể nảy sinh: Nếu lệnh được vận chuyển, nhưng ghi nợ lỗi? Điều gì nếu ghi nợ thành công, nhưng để không bao giờ được nhập vào hoặc vận chuyển? Nếu đây là một môi trường khép kín, sau đó giám sát xử lý giao dịch (TP) (như IBM CICS, Encina Transarc, hoặc BEA System Tuxedo) có thể đảm bảo rằng tất cả hoặc không có bước được hoàn tất và các hệ thống cuối cùng đạt được một trạng thái nhất quán.

Nhưng giả sử modem của người dùng bị ngắt kết nối ngay sau khi họ nhấp chuột vào OK. Việc đặt hàng có thành công hay không? Giả sử đường truyền ngắt trước phản hồi đến. Liệu người dùng có phải đặt hàng lại không? Các vấn đề cơ bản là giám sát xử lý giao dịch không thể đưa người dùng vào một trạng thái nhất quán. Người dùng là một phần của môi trường hệ thống phần mềm, mà môi trường này là mở vì nó có thể cung cấp cho bất kỳ người dùng nào.

Các giải pháp có thể cho một môi trường mở bao gồm:

- Ứng dụng máy chủ có thể gửi email về vấn đề tín dụng, hoặc phát hiện các giao dịch trùng lặp.
- Một Java applet tải về có thể đồng bộ hóa với máy chủ sau khi kết nối bị lỗi được tái lập và khôi phục lại giao dịch; applet có thể giao tiếp sử dụng HTTP, hoặc trực tiếp với các đối tượng máy chủ thông qua IIOP hoặc RMI.
- Nếu có quá nhiều đơn đặt hàng phải xử lý đồng bộ, chúng có thể được đặt trong một hàng đợi tin nhắn, quản lý bởi một máy chủ MOM (Message Oriented Middleware) và khách hàng sẽ được thông báo bằng email khi có giao dịch hoàn tất.

Email thường được sử dụng để mọi người giao tiếp với nhau, vì vậy với việc sử dụng email, máy chủ hành xử như một agent thông minh.

Chú ý rằng mặc dù các ví dụ trên xem xét một người dùng làm việc với một doanh nghiệp cụ thể, các vấn đề phát sinh ở dạng nghiêm trọng hơn trong môi trường B2B. Nếu cửa hàng máy ảnh nhỏ của chúng ta đã được coi như chỉ là một phần trong mạng lưới cung cấp lớn, sẽ không có hy vọng buộc các bên khác tiến hành các giao dịch nội bộ của chúng ta bằng bất cứ cách nào hay hội tụ một cách tin cậy về một trạng thái phù hợp trên toàn hệ thống . Các mô hình sâu hơn hơn về các giao dịch và các tiến trình kinh doanh là cần thiết để đảm bảo rằng các hành vi chính xác được thực hiện trong các trường hợp như vậy.

Đặc tả hiện tại cho dịch vụ Web không giải quyết các giao dịch hoặc chỉ ra một mô hình giao dịch. Tổ chức OASIS đang phát triển một đặc tả như vậy, nhưng quan điểm của hầu hết người thực hiện chính là SOAP sẽ quản lý những giao dịch bằng cách nào đó. Nếu không có sự hướng dẫn của một tiêu chuẩn hoặc một phương pháp từ thỏa thuận của các nhà cung cấp lớn, các giao dịch sẽ được thực hiện theo cách cơ học, do đó làm mất đi hy vọng cho khả năng tương tác và mở rộng.

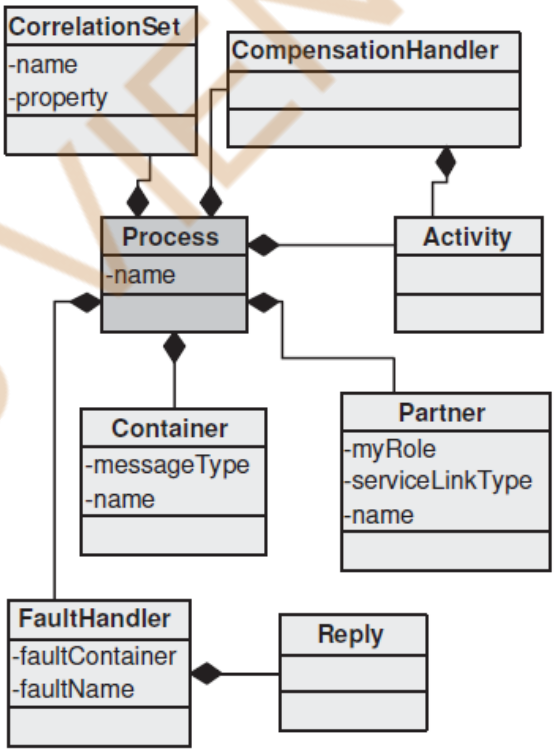
Một số vấn đề khác cho dịch vụ hợp bao gồm:

- An ninh sẽ khó khăn hơn, vì nhiều bên tham gia sẽ được tham gia và tính chất của các tương tác và nhu cầu của họ có thể là không trong dự kiến của các nhà thiết kế dịch vụ.

- Sẽ có sự không tương thích trong từ vựng, ngữ nghĩa, ngữ dụng (pragmatic) và giữa các bên cung cấp dịch vụ, môi giới dịch vụ, và yêu cầu dịch vụ.
- Do dịch vụ được hợp thành tự động, các vấn đề hiệu suất có thể phát sinh mà không được mong đợi.
- Thành phần dịch vụ năng động sẽ làm cho nó khó khăn để đảm bảo chất lượng dịch vụ (QoS) mà ứng dụng yêu cầu.

4.3.2 Công cụ BPEL cho hợp nghiệp vụ

BPEL (Business Process Execution Language) cho các dịch vụ Web (BPEL4WS) có thể sử dụng như là ngôn ngữ cài đặt cho các tiến trình thực thi và ngôn ngữ mô tả cho các giao thức kinh doanh không thực thi. Nó định nghĩa mô hình và ngữ pháp để mô tả cách thức các tương tác của dịch vụ Web trong các bên tham gia tiến trình, được gọi là các đối tác, phối hợp để đạt được mục tiêu kinh doanh, cũng như trạng thái và logic cần thiết cho sự phối hợp có thể xảy ra. Tương tác với từng đối tác thực hiện thông qua các giao diện dịch vụ Web cấp thấp hơn, như có thể được định nghĩa trong WSDL. BPEL có thể xác định các cơ chế để đối phó với các trường hợp ngoại lệ và lỗi xử lý, bao gồm làm thế nào các tiến trình đơn lẻ hoặc phức hợp sẽ được hỗ trợ khi xảy ra trường hợp ngoại lệ và lỗi hoặc khi một đối tác yêu cầu hủy bỏ. Hình 4.5 cho thấy các siêu mô hình cho BPEL4WS.



Hình 4.5: Siêu mô hình BPEL4WS

Một tài liệu BPEL4WS sử dụng XML để mô tả các khía cạnh sau đây của một tiến trình kinh doanh:

- Các đối tác: một danh sách các dịch vụ Web được gọi như là một phần của tiến trình.

- container: container dữ liệu được sử dụng bởi tiến trình này, cung cấp các định nghĩa của họ về các loại thông điệp WSDL. Container là cần thiết để lưu trữ dữ liệu trạng thái và lịch sử tiến trình dựa trên các thông điệp trao đổi giữa các tiến trình thành phần.
- variables: biến được sử dụng trong tiến trình này.
- faultHandlers: những bộ xử lý ngoại lệ.
- compensationHandler: hoàn tác khi một rollback giao dịch xảy ra.
- EventHandlers: xử lý các sự kiện bên ngoài (không đồng bộ).
- correlationSets: các ưu tiên và sự tương quan giữa các lời gọi dịch vụ Web mà không thể thể hiện như là một phần của logic tiến trình chính.
- logic tiến trình chính: một chuỗi các cấu trúc luồng điều khiển lồng nhau kết hợp các hoạt động nguyên thủy thành các thuật toán phức tạp. Các cấu trúc điều khiển bao gồm:

  - sequence, để thực hiện nối tiếp;

  - while, để thực hiện một vòng lặp;

  - switch, phân nhánh đa chiều;

  - pick, cho lựa chọn trong số những đường thay thế dựa trên một sự kiện bên ngoài;

  - flow, để thực thi song song. Trong các hoạt động thực hiện song song, những ràng buộc thứ tự thực hiện được chỉ ra bằng cách sử dụng các liên kết dịch vụ.

- Các cấu trúc điều khiển liên quan đến các hành động đơn nguyên (atomic action) sau đây:

  - invoke, gọi một dịch vụ Web cụ thể;

  - receive, một máy chủ đợi nhận một thông điệp từ một client sẽ gọi dịch vụ của máy chủ;

  - reply, sinh một trả lời cho một yêu cầu gọi;

  - wait, chờ đợi cho một thời hạn hoặc một khoảng thời gian;

  - assign, gán một giá trị cho một biến, giá trị này có thể đến từ một thông điệp nhận được;

  - throw, chỉ ra rằng một cái gì đó đã sai;

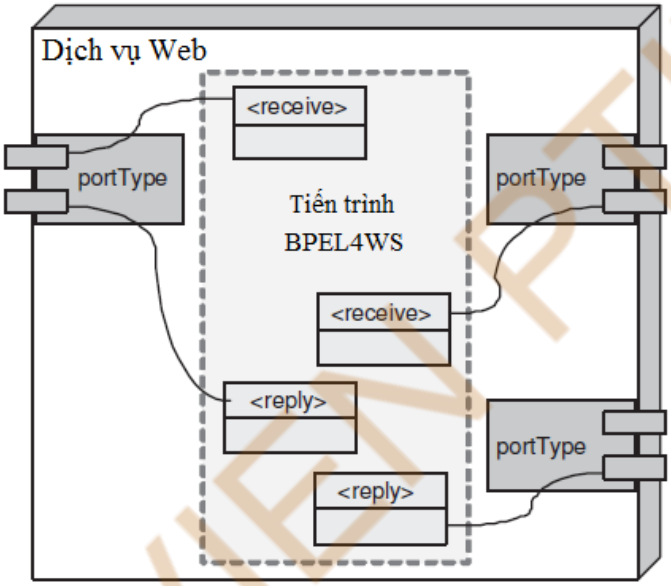
  - terminate, chấm dứt toàn bộ một thể hiện dịch vụ;

  - empty, không làm gì.

Khi mô hình hóa một giao thức kinh doanh bởi một tiến trình trừu tượng, BPEL4WS chỉ mô tả các khía cạnh công khai về giao thức. Ví dụ, trong một giao thức chuỗi cung ứng, BPEL4WS sẽ mô tả vai trò của người mua và người bán như các tiến trình trừu tượng, với mối quan hệ được mô hình như một liên kết dịch vụ. Các tiến trình trừu tượng bị hạn chế đến việc xử lý các giá trị chứa trong các thuộc tính thông điệp, và sử dụng các giá trị không xác định để phản ánh kết quả của hành vi riêng tư ẩn.

Khi mô hình hóa một tiến trình kinh doanh thực thi, BPEL4WS không nhất thiết phải mô tả một cài đặt riêng lẻ của một đối tác một cách triệt để, nhưng nó phải mô tả một định dạng thực thi khả chuyển cho các tiến trình kinh doanh. Những tiến trình đó thực thi và tương tác với các đối tác của chúng một cách phù hợp không phụ thuộc vào nền tảng hỗ trợ hoặc các mô hình lập trình được sử dụng bởi một cài đặt cụ thể.

Kết quả của việc sử dụng BPEL4WS để mô hình một tiến trình kinh doanh thực thi là một dịch vụ Web mới bao gồm các dịch vụ hiện có. Giao diện của dịch vụ tổng hợp là một tập WSDL portTypes, giống như các dịch vụ Web khác. Hình 4.6 minh họa cái nhìn từ bên ngoài của một quá trình BPEL4WS.



Hình 4.6: Một tiến trình BPEL4WS là một dịch vụ Web phức hợp với một giao diện bao gồm các WSDL portType như các dịch vụ Web khác

4.3.2.1 Luồng giao dịch

BPEL4WS cung cấp một giao thức bồi thường. Nó cho phép điều khiển linh hoạt cho rollback và đảo chiều thông qua các định nghĩa cụ thể cho ứng dụng để xử lý lỗi và bồi thường, và kết quả là ta có LRT (Long-Running (Business) Transaction).

Một LRT có thể được hoàn tác bằng cách đảo ngược các hoạt động riêng rẽ, sử dụng các quy tắc kinh doanh hay phải phụ thuộc vào ứng dụng. Các phần tử phạm vi mô tả các bộ phận của một hành vi được phép đảo ngược bởi một bộ xử lý bồi thường. Các phạm vi có thể được lồng nhau với một độ sâu tùy ý.

Một LRT xảy ra trong một thể hiện tiến trình kinh doanh đơn lẻ, và không có sự phối hợp phân tán giữa các đối tác liên quan đến một kết quả đã được thỏa thuận. Việc làm thế nào để đạt được thỏa thuận phân tán nằm ngoài phạm vi của BPEL4WS.

4.3.2.2 Cài đặt một dịch vụ Web BPEL4WS

Một tiến trình BPEL4WS là một thuật toán thể hiện như một biểu đồ luồng, trong đó mỗi bước là một hoạt động. Thông tin được truyền giữa các hoạt động thông qua các container

dữ liệu và sử dụng các lệnh <assign>. Ví dụ, địa chỉ của khách hàng sẽ được sao chép từ một đơn đặt hàng tới một yêu cầu vận chuyển bằng cách sau đây:

```
<assign>
<copy>
<from c o n t a i n e r ="PO" p a r t ="customerAddress"/>
<to c o n t a i n e r ="shippingRequest" p a r t ="customerInfo"/>
</copy>
</ assign>
```

Hình 4.7: Ví dụ về một yêu cầu vận chuyển

Một liên kết dịch vụ được sử dụng để xác định các mối quan hệ giữa hai đối tác và vai trò của mỗi đối tác thực hiện. Ví dụ, một liên kết dịch vụ giữa người mua và người bán có thể là:

```
<serviceLinkType name="BuySellLink"
xmlns="http://schemas.xmlsoap.org/ws/2002/07/service-link/">
<r o l e name="Buyer">
<portType name="BuyerPortType"/>
</ r o l e>
<r o l e name="Seller">
<portType name="SellerPortType"/>
</ r o l e>
</ serviceLinkType>
```

Hình 4.8: Ví dụ về một liên kết giữa người mua và người bán

Sau đây là một ví dụ hoàn chỉnh của một tiến trình BPEL4WS thực thi để thực hiện một dịch vụ thông báo chứng khoán:

```
<!ENTITY BPEL
"http://schemas.xmlsoap.org/ws/2002/07/business-process"
<p r o c e s s name="simple"
targetNamespace="urn:simple:stockQuoteService"
x m l n s : t n s="urn:simple:stockQuoteService"
xmlns:sqp="http://tempuri.org/services/stockquote"
xmlns=&BPEL;/>
<c o n t a i n e r s >
<c o n t a i n e r name="request"
messageType="tns:request"/>
<c o n t a i n e r name="response"
messageType="tns:response"/>
<c o n t a i n e r name="invocationRequest"
messageType="sqp:GetQInput"/>
<c o n t a i n e r name="invocationResponse"
messageType="sqp:GetQOutput"/>
</ c o n t a i n e r s >
<p a r t n e r s >
<p a r t n e r name="caller"
serviceLinkType="tns:StockQuoteSLT"/>
<p a r t n e r name="provider"
serviceLinkType="tns:StockQuoteSLT"/>
</ p a r t n e r s >
<sequence name="sequence">
```

```
<r e c e i v e name="receive" p a r t n e r ="caller"
portType="tns:StockQuotePT"
o p e r a t i o n="wantQuote" c o n t a i n e r ="request"
c r e a t e I n s t a n c e ="yes"/>
<assign>
<copy>
<from c o n t a i n e r ="request" p a r t ="symbol"/>
<to c o n t a i n e r ="invocationRequest" p a r t ="symbol"/>
</copy>
</ assign>
<invoke name="invoke" p a r t n e r ="provider"
portType="sqp:StockQuotePT"
o p e r a t i o n="getQuote"
i n p u t C o n t a i n e r="invocationRequest"
o u t p u t C o n t a i n e r="invocationResponse"/>
<assign>
<copy>
<from c o n t a i n e r ="invocationResponse" p a r t ="quote"/>
<to c o n t a i n e r ="response" p a r t ="quote"/>
</copy>
</ assign>
<r e p l y name="reply" p a r t n e r ="caller"
portType="tns:StockQuotePT"
o p e r a t i o n="wantQuote" c o n t a i n e r ="response"/>
</ sequence>
</ process>
```

Hình 4.8: Ví dụ về thực hiện tra cứu thông tin chứng khoán

Tiến trình này là một chuỗi năm bước đơn giản mà bắt đầu khi một yêu cầu cho một thông báo nhận được từ bên gọi. Các yêu cầu được sao chép vào một container gọi, hoạt động getQuote được gọi với các thông số của yêu cầu, kết quả sẽ được sao chép vào một container kết quả, và một thông báo trả lời được trả về cho bên yêu cầu.

4.3.2.3 Case study: xây dựng kiến trúc hướng dịch vụ và sử dụng công nghệ BPEL cho hợp dịch vụ

Xây dựng một mô tả BPEL4WS cho một tiến trình mà bạn đã quen thuộc, chẳng hạn như khi rút tiền từ máy ATM, trả tiền mua hàng bằng thẻ tín dụng, đăng ký lớp học, học phí và lệ phí, hoặc xin visa sinh viên hay visa du lịch. Hãy tưởng tượng rằng mỗi tiến trình là một tương tác hai phía: bạn và những người tham gia khác tương ứng (ngân hàng, người buôn bán, bộ phận đăng ký của trường đại học, phòng kế toán trường đại học, lãnh sự quán nước ngoài).

4.4 BÀI TẬP

1. Trình bầy vai trò của hợp dịch vụ?
2. Trình bầy các thành phần của các công cụ BPEL cho hợp nghiệp vụ?
3. Xây dựng một mô tả BPEL4WS cho một tiến trình tra cứu điểm các môn học theo một học kỳ của sinh viên (xem bài tập 1 chương 3);

CHƯƠNG 5 CÁC MÔ HÌNH KIẾN TRÚC

5.1 KIẾN TRÚC HƯỚNG DỊCH VỤ

Phần trên đã mô tả các trường hợp sử dụng cùng với những yêu cầu phức tạp đối với các phương pháp tính toán. Các yêu cầu này có thể được đáp ứng dễ dàng hơn thông qua một kiến trúc phù hợp với các đặc tính thiết yếu của các trường hợp sử dụng ở trên mà người ta gọi là kiến trúc hướng dịch vụ (SOA – Service Oriented Architecture).

5.1.1 Các yếu tố của kiến trúc hướng dịch vụ

Để đạt được các ưu điểm trên, SOA quy định các yêu cầu sau:

Kết nối lỏng (Loose coupling) – Không có thuộc tính giao dịch chặt chẽ được áp dụng giữa các thành phần. Nói chung, việc xác định tính nhất quán của dữ liệu là không phù hợp qua các nguồn thông tin từ các bộ phận của các thành phần khác nhau. Tuy nhiên, điều này khá hợp lý cho các mối quan hệ hợp đồng mức cao mà qua đó các tương tác giữa các thành phần được quy định.

Tính trung lập cài đặt (Implementation neutrality) - Giao diện là quan trọng. Chúng ta không thể phụ thuộc vào các chi tiết của việc triển khai các thành phần tương tác. Đặc biệt, phương pháp này không thể cụ thể cho các ngôn ngữ lập trình.

Khả năng cấu hình linh hoạt (Flexible configurability) - Hệ thống được cấu hình trễ và linh hoạt. Nói cách khác, các thành phần khác nhau được ràng buộc với nhau trễ trong quá trình này. Các cấu hình có thể tự động thay đổi.

Thời gian hoạt động dài (Long lifetime) - Không nhất thiết phải cần các thành phần có thời gian sống rất lâu. Tuy nhiên, vì chúng ta đang xử lý các tính toán giữa các thành phần không đồng nhất và tự trị trong môi trường động nên luôn cần đến khả năng xử lý các trường hợp ngoại lệ. Điều này có nghĩa là các thành phần phải tồn tại đủ lâu để có thể phát hiện bất kỳ trường hợp ngoại lệ nào có liên quan, để có hành động sửa sai, và để đáp ứng với các hành động khắc phục được thực hiện bởi những thành phần khác. Các thành phần phải tồn tại đủ lâu để được khám phá, được dựa vào, và để tạo ra niềm tin trong hành vi.

Mức độ chi tiết (Granularity) – Các bên tham gia một SOA nên được hiểu ở mức thô. Đó là, thay vì mô hình hóa các hành động và tương tác ở một mức độ chi tiết, sẽ tốt hơn để nắm bắt những giá trị cao cấp cần thiết có thể cho thấy mục đích của hợp đồng kinh doanh giữa các thành viên tham gia. Mức thô giúp giảm sự phụ thuộc giữa các thành viên tham gia và làm giảm thông tin cần liên lạc, chỉ cần tập trung với một vài thông điệp có ý nghĩa.

Các nhóm (teams) - Thay vì đóng khung các tính toán một cách tập trung, sẽ tốt hơn khi suy nghĩ về cách thức tính toán được thực hiện bởi các bên tự trị. Nói cách khác, thay vì một bên tham gia chỉ huy các đối tác của mình, tính toán trong hệ thống mở nên là một vấn đề của các đối tác kinh doanh làm việc như một nhóm. Đó là, thay vì một bên riêng lẻ, một nhóm các bên tham gia hợp tác là một mô hình tốt hơn.

5.1.2 So sánh lời gọi thủ tục từ xa (RPC) với định hướng tài liệu (OD)

Có hai khung nhìn chính về dịch vụ Web. Các dịch vụ có thể được hiểu theo khung nhìn RPC làm trung tâm hoặc tài liệu làm trung tâm. Khung nhìn thứ nhất xem dịch vụ là việc đưa ra một tập các phương pháp được gọi từ xa, nghĩa là, thông qua các cuộc gọi thủ tục từ xa. Khung nhìn sau coi dịch vụ là việc trao đổi tài liệu với nhau. Trong cả hai khung nhìn, những gì được truyền đi là các tài liệu XML và những gì được tính toán là những đối tượng dựa trên hoặc tương ứng với các tài liệu XML. Tuy nhiên, có một sự khác biệt quan trọng về khái niệm.

Khung nhìn RPC xem các tài liệu XML liên quan đến việc tính toán phân phối tổng thể. Các tài liệu chỉ đơn thuần là biểu diễn tuần tự (serialization) của các đối tượng kinh doanh dùng để tính toán. Khung nhìn coi tài liệu làm trung tâm coi tài liệu như biểu diễn (representation) chính và mục đích chính của tính toán phân tán. Mỗi thành phần sẽ đọc, tạo, lưu, và truyền các tài liệu. Các tài liệu được tạm thời hiện thực hóa thành các đối tượng kinh doanh để cho phép tính toán.

Khung nhìn RPC do đó tương ứng với dịch vụ một tấm gỗ dán mỏng của dịch vụ Web trên một ứng dụng hiện có. Ứng dụng này sẽ quyết định những tính năng của dịch vụ sẽ hỗ trợ. Khung nhìn tài liệu xem xét các dịch vụ Web tự nhiên hơn, coi đó như một phương tiện thực hiện các mối quan hệ kinh doanh. Các văn bản phải được xử lý (và các mối quan hệ của chúng) xác định các chức năng của các dịch vụ. Các đối tượng kinh doanh không được lộ ra cho phía bên kia biết.

Vì lý do này, khung nhìn tài liệu làm trung tâm hợp lý hơn với các trường hợp sử dụng chính của chúng ta trong việc sử dụng các dịch vụ trong môi trường mở. Khung nhìn RPC hợp lý hơn cho trường hợp sử dụng làm cho các ứng dụng đã phát triển liên tác độc lập. Những gì xảy ra là các nhà phát triển ứng dụng thể hiện giao diện ứng dụng của họ trong các hình thức dịch vụ Web, rồi sau đó có thể bị ràng buộc theo cách thông thường. Nếu các ứng dụng được thiết kế hỗ trợ tích hợp phương thức, thì khung nhìn RPC của dịch vụ là hợp lý hơn cho liên tác như vậy. Tuy nhiên, nếu các ứng dụng được thiết kế để hoạt động như các thành phần độc lập, thì khung nhìn tài liệu sẽ hợp lý hơn, ngay cả đối với liên tác ứng dụng.

5.2 KIẾN TRÚC RESTful

5.2.1 Khái niệm

Thuật ngữ REST xuất phát từ luận án Tiến sĩ của Roy Fielding, được xuất bản năm 2000, và là viết tắt của từ ***REpresentational State Transfer***. REST tự nó không phải là một kiến trúc; REST là một tập hợp các ràng buộc, khi được áp dụng để thiết kế một hệ thống, sẽ tạo ra một kiểu kiến trúc phần mềm. Nếu thực hiện tất cả các nguyên tắc REST, ta sẽ tạo ra một hệ thống có các vai trò cụ thể đối với dữ liệu, thành phần, siêu liên kết, giao thức truyền thông và người tiêu dùng dữ liệu. Một hệ thống RESTful sẽ tuân thủ các ràng buộc sau:

- là một hệ thống máy khách-máy chủ
- stateless — dịch vụ không cần lưu trữ phiên người dùng; nói cách khác, mỗi yêu cầu phải độc lập với những yêu cầu khác

- phải hỗ trợ hệ thống bộ nhớ đệm — cơ sở hạ tầng mạng phải hỗ trợ bộ nhớ đệm ở các cấp độ khác nhau
- phải đáp ứng truy cập thống nhất — mỗi tài nguyên phải có một địa chỉ duy nhất và một điểm truy cập hợp lệ
- phải được phân lớp — hỗ trợ khả năng mở rộng
- phải cung cấp mã theo yêu cầu — mặc dù đây là một ràng buộc tùy chọn, các ứng dụng có thể được mở rộng trong thời gian chạy bằng cách cho phép tải xuống mã theo yêu cầu, ví dụ: Java Applet

Những ràng buộc này không quy định loại công nghệ sẽ sử dụng; nó chỉ xác định cách dữ liệu được truyền giữa các thành phần và lợi ích của việc tuân theo các hướng dẫn. Do đó, một hệ thống RESTful có thể được thực hiện trong bất kỳ kiến trúc mạng có sẵn. Quan trọng hơn, chúng ta không cần phải phát minh ra công nghệ hoặc giao thức mạng mới: chúng ta có thể sử dụng các cơ sở hạ tầng mạng hiện có như Web để tạo ra các kiến trúc RESTful. Do đó, kiến trúc RESTful là kiến trúc có thể bảo trì, mở rộng và phân phối.

Web là một hệ thống điển hình của RESTful, như vậy tai sao chúng ta lại đưa các yêu cầu RESTful vào phát triển ứng dụng web khi chính bản thân Web đã tuân thủ theo kiến trúc RESTful.

Trước tiên, chúng ta cần phải xác định điều kiện đủ để Web trở nên RESTful. Về cơ bản, web tĩnh là RESTful, vì các trang web tĩnh tuân theo định nghĩa của Fielding về kiến trúc RESTful. Ví dụ: cơ sở hạ tầng web hiện có cung cấp hệ thống bộ nhớ đệm, kết nối không trạng thái và siêu liên kết duy nhất đến tài nguyên, nơi tài nguyên là tất cả các tài liệu có sẵn trên mọi trang web và phần trình bày của các tài liệu này đã được thiết lập bởi các tệp có thể đọc được trên trình duyệt (tệp HTML, cho thí dụ). Do đó, web tĩnh là một hệ thống được xây dựng theo kiểu kiến trúc giống REST.

Tuy nhiên, các ứng dụng web động truyền thống không phải lúc nào cũng RESTful, bởi vì chúng thường phá vỡ một số ràng buộc trên. Ví dụ: hầu hết các ứng dụng động không đáp ứng stateless, vì máy chủ yêu cầu theo dõi người dùng thông qua các bộ chưa phiên hoặc cookie phía máy khách. Do đó, có thể kết luận rằng web động thường không được xây dựng theo kiểu kiến trúc giống REST.

Trong phần tiếp theo, các yếu tố trừu tượng tạo nên một hệ thống RESTful sẽ được giới thiệu, cụ thể là tài nguyên, biểu diễn, URI và các loại yêu cầu HTTP tạo nên giao diện thống nhất được sử dụng để truyền dữ liệu máy khách / máy chủ.

5.2.2 Tài nguyên và biểu diễn tài nguyên

Tài nguyên RESTful là bất cứ thứ gì có thể được đánh địa chỉ trên Web. Tức là là các tài nguyên có thể được truy cập và chuyển giữa các máy khách và máy chủ. Điều đó dẫn đến tài nguyên là một ánh xạ lô gíc, theo thời gian tới một khái niệm trong miền bài toán đang triển khai giải pháp.

Dưới đây là một số ví dụ về tài nguyên REST:

- Một câu chuyện tin tức

- Nhiệt độ ở NY lúc 4:00 chiều EST
- Tờ khai thuế được lưu trữ trong cơ sở dữ liệu IRS
- Danh sách lịch sử sửa đổi mã trong kho lưu trữ như SVN hoặc CVS
- Một học sinh trong lớp học ở trường nào đó
- Kết quả tìm kiếm cho một mục cụ thể trong chỉ mục web, chẳng hạn như Google

Mặc dù ánh xạ của tài nguyên là duy nhất, trong một số trường hợp các yêu cầu khác nhau đối với một tài nguyên có thể trả về cùng một biểu diễn nhị phân được lưu trữ trong máy chủ. Ví dụ: giả sử chúng ta có một tài nguyên trong ngữ cảnh của một hệ thống xuất bản. Sau đó, yêu cầu "bản sửa đổi mới nhất được xuất bản" và yêu cầu "bản sửa đổi số 12" tại một thời điểm nào đó sẽ trả về cùng một đại diện của tài nguyên: bản sửa đổi cuối cùng là bản sửa đổi 12. Tuy nhiên, khi bản sửa đổi mới nhất được xuất bản được tăng lên đến phiên bản 13, yêu cầu đối với bản sửa đổi mới nhất sẽ trả về phiên bản 13 và yêu cầu về bản sửa đổi 12 sẽ tiếp tục trả về phiên bản 12. Vì là tài nguyên trong kiến trúc RESTful, mỗi tài nguyên này có thể được truy cập trực tiếp và độc lập, nhưng các yêu cầu khác nhau có thể trỏ với cùng một dữ liệu.

Bởi vì HTTP được sử dụng để giao tiếp, ta có thể chuyển bất kỳ loại thông tin nào có thể được truyền giữa máy khách và máy chủ. Ví dụ: nếu yêu cầu tệp văn bản từ Vnexpress, trình duyệt của sẽ nhận được tệp văn bản. Nếu yêu cầu một bộ phim Flash từ YouTube, trình duyệt của sẽ nhận được một bộ phim Flash. Dữ liệu được truyền trực tuyến trong cả hai trường hợp qua TCP / IP và trình duyệt biết cách diễn giải các luồng nhị phân thông qua trường Content-Type trong phần Header của HTTP. Do đó, trong hệ thống RESTful, việc biểu diễn tài nguyên phụ thuộc vào kiểu mong muốn của người gọi (kiểu MIME), được chỉ định trong yêu cầu của giao thức truyền thông.

Biểu diễn của tài nguyên là những gì được gửi qua lại giữa các máy khách và máy chủ. Biểu diễn là trạng thái tạm thời của dữ liệu thực tế nằm trong một số thiết bị lưu trữ tại thời điểm được yêu cầu. Nói chung, đó là một luồng nhị phân cùng với siêu dữ liệu của nó mô tả cách mà máy khách hoặc máy chủ sử dụng luồng (siêu dữ liệu cũng có thể chứa thông tin bổ sung về tài nguyên, ví dụ: xác thực, thông tin mã hóa hoặc mã bổ sung được thực thi trong thời gian chạy).

Trong suốt vòng đời của dịch vụ web, có thể có nhiều khách hàng yêu cầu tài nguyên. Các máy khách khác nhau có thể sử dụng các đại diện khác nhau của cùng một tài nguyên. Do đó, biểu diễn có thể có nhiều dạng khác nhau, chẳng hạn như hình ảnh, tệp văn bản hoặc luồng XML hoặc luồng JSON, nhưng phải có sẵn thông qua cùng một URI.

Đối với các yêu cầu do con người tạo thông qua trình duyệt web, biểu diễn thường ở dạng trang HTML. Đối với các yêu cầu tự động từ các dịch vụ web khác, khả năng đọc không quan trọng bằng và có thể sử dụng một biểu diễn hiệu quả hơn như XML.

5.2.3 URI

Mã định danh tài nguyên đồng nhất, hoặc Uniform Resource Identifier - URI, trong dịch vụ web RESTful là một siêu liên kết đến tài nguyên và là phương tiện duy nhất để máy khách và máy chủ trao đổi các biểu diễn.

Các ràng buộc RESTful không quy định các URI phải là siêu liên kết. Ở đây RESTful URI là siêu liên kết, bởi vì ta đang sử dụng Web để tạo các dịch vụ web. Nếu sử dụng một bộ công nghệ hỗ trợ khác, URI RESTful sẽ khác.

Trong hệ thống RESTful, URI không thay đổi theo thời gian, vì bản chất của kiến trúc này là để quản lý các dịch vụ, định vị tài nguyên, dàn xếp các biểu diễn và sau đó gửi lại phản hồi với các tài nguyên được yêu cầu. Quan trọng hơn, nếu ta thay đổi cấu trúc của thiết bị lưu trữ ở cấp máy chủ (ví dụ: hoán đổi máy chủ cơ sở dữ liệu), thì các URI sẽ vẫn giữ nguyên và còn nguyên giá trị, miễn là dịch vụ web trực tuyến hoặc ngữ cảnh của tài nguyên không thay đổi.

Không có ràng buộc REST, tài nguyên được truy cập qua vị trí: địa chỉ web điển hình là các URI cố định. Ví dụ: nếu ta đổi tên tệp trên máy chủ web, URI sẽ khác; nếu ta di chuyển tệp sang cây thư mục khác trong máy chủ web, URI sẽ thay đổi. Lưu ý rằng ta có thể sửa đổi máy chủ web của mình để thực hiện chuyển hướng trong thời gian chạy nhằm duy trì khả năng định địa chỉ, nhưng nếu thay đổi tất cả tệp, các quy tắc sẽ không thể quản lý được.

5.2.4 Giao diện đồng nhất qua yêu cầu HTTP

Trong các phần trước, các khái niệm về tài nguyên và biểu diễn tài nguyên đã được giới thiệu. Trong đó, tài nguyên là ánh xạ các trạng thái thực thể được trao đổi giữa máy khách và máy chủ. Ngoài ra, các biểu diễn của tài nguyên được vận chuyển qua lại giữa máy khách và máy chủ thông qua giao thức truyền thông trong thời gian chạy —qua HTTP. Trong phần này, ý nghĩa của việc trao đổi các biểu diễn này và ý nghĩa của việc các máy khách và máy chủ thực hiện các hành động trên các tài nguyên này sẽ được xem xét chi tiết.

Việc phát triển các dịch vụ web RESTful tương tự như những gì đã làm để phát triển một ứng dụng web điển hình. Tuy nhiên, sự khác biệt cơ bản giữa phát triển ứng dụng web hiện đại và truyền thống là cách các hành động được thực hiện trên dữ liệu trừu tượng. Cụ thể, phương pháp phát triển hiện đại bắt nguồn trong ngữ nghĩa của danh từ (trao đổi tài nguyên); trong khi kiểu cũ được bắt nguồn trong ngữ nghĩa của động từ (hành động từ xa được thực hiện trên dữ liệu). Phương pháp hiện đại triển khai một dịch vụ web RESTful; còn kiểu cũ triển khai một dịch vụ tương tự RPC (lời thủ tục từ xa). Ngoài ra, một dịch vụ RESTful sửa đổi trạng thái của dữ liệu thông qua biểu diễn của tài nguyên; còn một dịch vụ RPC ẩn phần biểu diễn dữ liệu và thay vào đó gửi các lệnh để sửa đổi trạng thái của dữ liệu ở cấp độ máy chủ (không bao giờ biết dữ liệu trông như thế nào). Cuối cùng, trong phát triển ứng dụng web hiện đại, sự mơ hồ về thiết kế và triển khai được hạn chế, bởi vì chỉ có bốn hành động có thể thực hiện đối với tài nguyên — Tạo, Truy xuất, Cập nhật và Xóa (CRUD). Còn trong phát triển ứng dụng web truyền thống, có thể có vô số hành động không tên hoặc không có tiêu chuẩn triển khai.

Do đó, với các vai trò được phân định cho tài nguyên và biểu diễn tài nguyên, các hành động CRUD có thể được ánh xạ với các phương thức HTTP POST, GET, PUT và DELETE như sau:

Hành động	Giao thức HTTP tương đương
TẠO	POST

LẤY	GET
CẬP NHẬT	PUT
XÓA	DELETE

Ở dạng đơn giản nhất, các dịch vụ web RESTful là các ứng dụng được nối mạng điều khiển trạng thái của tài nguyên. Trong ngữ cảnh này, thao tác tài nguyên có nghĩa là tạo, truy xuất, cập nhật và xóa tài nguyên. Tuy nhiên, các dịch vụ web RESTful không chỉ giới hạn trong bốn khái niệm thao tác dữ liệu cơ bản này. Ngược lại, các dịch vụ web RESTful có thể thực thi logic ở cấp máy chủ, nhưng cần nhớ rằng mọi kết quả phải là biểu diễn tài nguyên.

Phần tiếp theo sẽ chi tiết hóa 4 hành động và giao thức HTTP tương ứng.

5.2.4.1 Phương thức GET – Nhận Tài nguyên

Phương thức GET được sử dụng để NHẬN tài nguyên.

Trước khi đi sâu vào cơ chế của yêu cầu HTTP GET r equest, ta cần xác định tài nguyên là gì trong ngữ cảnh dịch vụ web và loại biểu diễn mà chúng ta đang trao đổi.

Phần tiếp sẽ sử dụng ví dụ về dịch vụ web với các tác vụ liên quan đến sinh viên trong lớp học, tại vị trí là http://restfuljava.com/. Giả sử ta có biểu diễn XML của một sinh viên như sau:

```
  <student>
     <name>Jane</name>
     <age>10</age>
     <link>/students/Jane</link>
</student>
And a list of students to look like:
  <students>
     <student>
        <name>Jane</name>
        <age>10</age>
        <link>/students/Jane</link>
   </student>
<student>
   <name>John</name>
   <age>11</age>
         <link>/students/John</link>
               </student>
               <link>/students</link>
            </students>
```

Giả sử URI có dạng http://restfuljava.com/students sẽ cho phép truy cập danh sách các sinh viên và http://restfuljava.com/students/{name} để truy cập một sinh viên cụ thể có định danh duy nhất thông qua giá trị name.

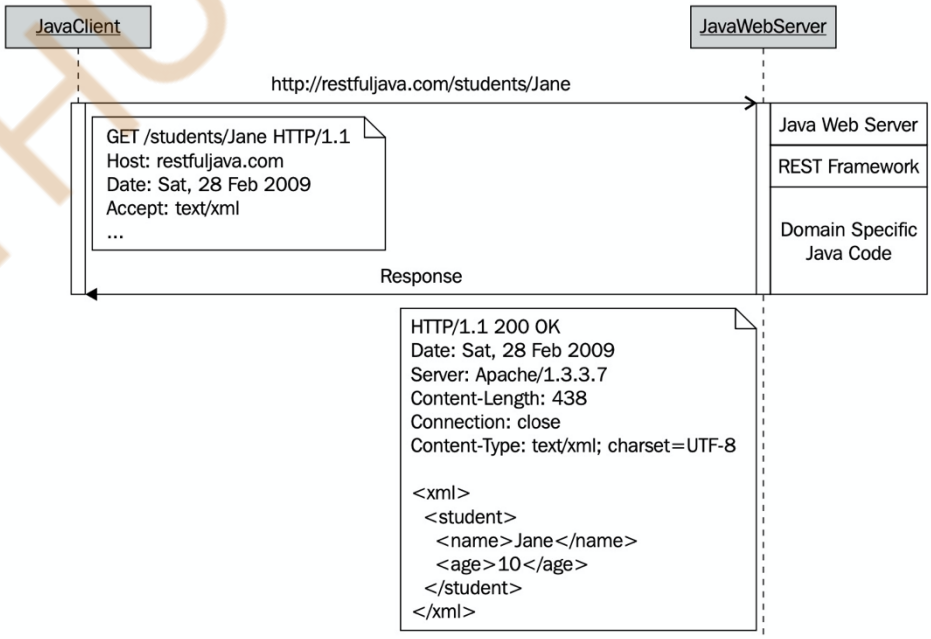
Bây giờ ta có thể gửi yêu cầu tới dịch vụ web. Ví dụ: nếu muốn hồ sơ một sinh viên có tên Jane, yêu cầu sẽ được gửi tới URI http://restfuljava.com/students/Jane. Biểu diễn của Jane, tại thời điểm yêu cầu, có thể như sau:

```
<student>
    <name>Jane</name>
    <age>10</age>
    <link>/students/Jane</link>
</student>
```

Tương tự, danh sách sinh viên có thể được truy cập thông qua URI http://restfuljava.com/students. Phản hồi từ dịch vụ sẽ bao gồm biểu diễn của tất cả sinh viên có thể như sau (giả sử có hai sinh viên có sẵn):

```
<students>
    <student>
        <name>Jane</name>
        <age>10</age>
        <link>/students/Jane</link>
    </student>
    <student>
        <name>John</name>
        <age>11</age>
        <link>/students/John</link>
</student>
    <link>/students</link>
</students>
```

Đi sâu vào chi tiết của yêu cầu ta thấy, yêu cầu truy xuất tài nguyên Jane sử dụng phương thức GET với URI http://restfuljava.com/students/Jane. Biểu đồ trình tự của yêu cầu GET của ta trông như sau:



Hình 5-1. Sơ đồ tương tác sử dụng phương thức GET

Trình tự được thực hiên bao gồm:

1) Ứng dụng khách Java đưa ra một yêu cầu HTTP với phương thức GET và Jane là định danh cho sinh viên.
2) Máy khách đặt kiểu biểu diễn mà nó có thể xử lý thông qua trường Accept trong tiêu đề yêu cầu.
3) Máy chủ web nhận và diễn giải yêu cầu GET là hành động truy xuất tài nguyên. Tại thời điểm này, máy chủ web chuyển quyền kiểm soát cho nền tảng RESTful để xử lý yêu cầu. Lưu ý rằng nền tảng RESTful không tự động truy xuất tài nguyên. Nền tảng RESTful chỉ hỗ trợ làm dễ dàng việc thực hiện các ràng buộc REST. Logic nghiệp vụ và thực thi việc lưu trữ là vai trò của mã Java.
4) Chương trình phía máy chủ tìm kiếm tài nguyên Jane. Tìm tài nguyên có thể là tìm trong cơ sở dữ liệu, hệ thống tệp hoặc lệnh gọi đến một dịch vụ web khác.
5) Khi chương trình tìm thấy Jane, nó sẽ chuyển đổi dữ liệu nhị phân của tài nguyên thành biểu diễn yêu cầu của khách hàng.
6) Với biểu diễn được chuyển đổi sang XML, máy chủ sẽ gửi lại phản hồi HTTP với mã số 200 cùng với biểu diễn XML là tải trọng chính. Lưu ý rằng nếu có bất kỳ lỗi nào, máy chủ HTTP sẽ báo cáo lại mã lỗi thích hợp.

Tất cả các thông điệp giữa máy khách và máy chủ là lời gọi theo giao thức HTTP tiêu chuẩn. Đối với mỗi hành động truy xuất, một yêu cầu GET được gửi đi và client nhận được phản hồi HTTP trở lại với tải trọng của phản hồi là biểu diễn của tài nguyên hoặc nếu có lỗi, mã lỗi HTTP tương ứng (ví dụ: 404 nếu là tài nguyên không tìm thấy; 500, nếu có vấn đề với mã Java ở dạng ngoại lệ).

Lấy về biểu diễn tất cả sinh viên hoạt động giống như cách lấy biểu diễn cho một sinh viên riêng lẻ, mặc dù bây giờ ta sử dụng URI http://restfuljava.com/students và kết quả là biểu diễn XML:

```
<students>
    <student>
        <name>Jane</name>
        <age>10</age>
        <link>/students/Jane</link>
    </student>
    <student>
        <name>John</name>
        <age>11</age>
        <link>/students/John</link>
```

```
</student>
    <link>/students</link>
</students>
```

Phương thức HTTP GET chỉ nên được sử dụng để truy xuất các biểu diễn của tài nguyên. Mặc dù yêu cầu GET có thể được sử dụng để cập nhật trạng thái dữ liệu trong máy chủ, nhưng điều này không được khuyến khích. Yêu cầu GET phải an toàn và hợp lý.

Để một yêu cầu được an toàn, tức là nhiều yêu cầu đến cùng một tài nguyên không thay đổi trạng thái của dữ liệu trong máy chủ. Giả sử chúng ta có một biểu diễn R và các yêu cầu xảy ra tại thời điểm t. Sau đó, một yêu cầu tại thời điểm t1 cho tài nguyên R trả về R1; sau đó, một yêu cầu tại thời điểm t2 cho tài nguyên R trả về R2; miễn là không có hành động cập nhật nào được thực hiện trong khoảng thời gian từ t1 đến t2, khi đó R1 = R2 = R.

5.2.4.2 Phương thức POST – Tạo tài nguyên

Các phương thức POST được sử dụng để TẠO tài nguyên.

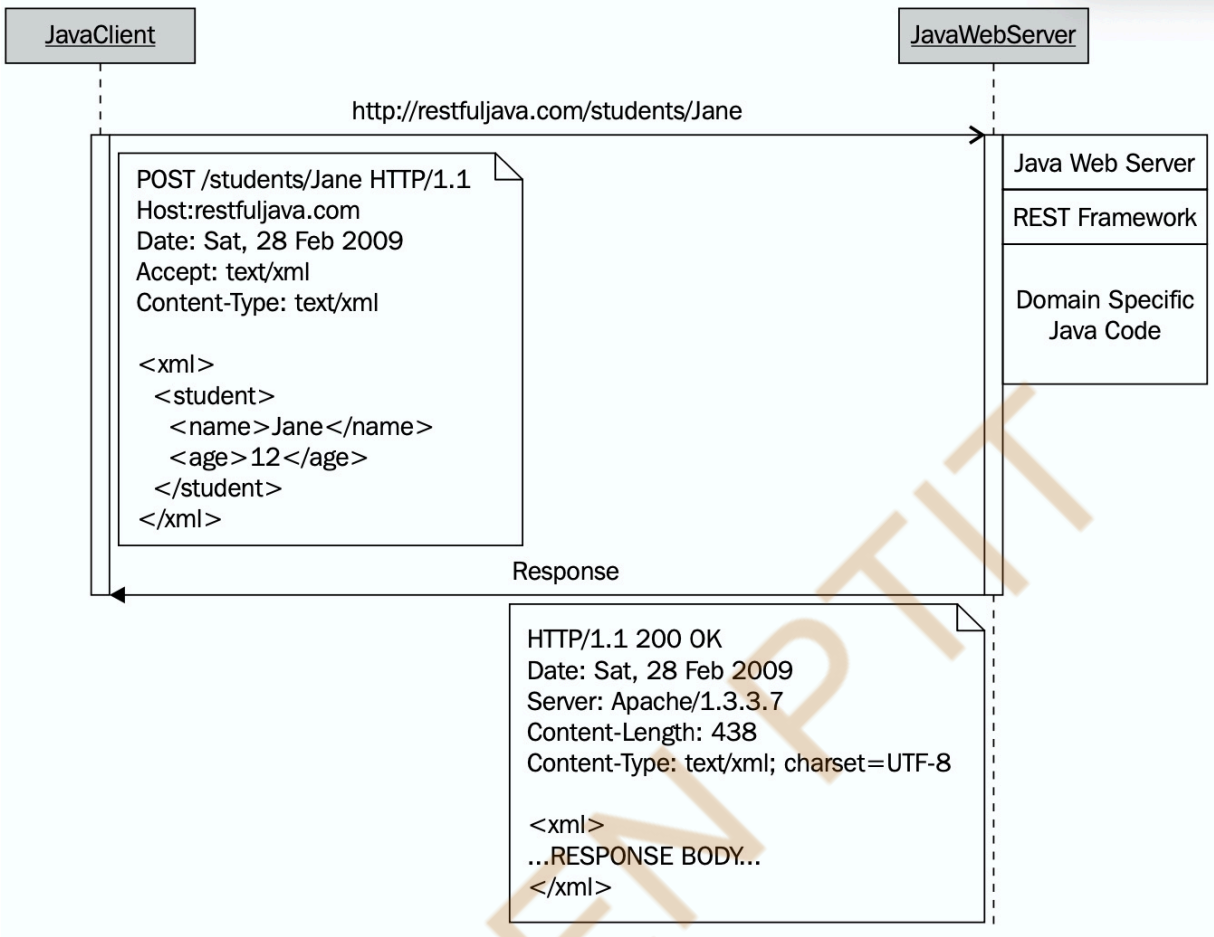
Để tạo mới một sinh viên, ta sử dụng phương thức HTTP POST. URI để tạo sinh viên mới trong danh sách sẽ là http://restfuljava.com/students/Jane. Loại phương thức cho yêu cầu được thiết lập bởi máy khách.

Giả sử Jane không tồn tại trong danh sách đã có. Biểu diễn XML mới của Jane được thể hiện như sau:

```
  <student>
      <name>Jane</name>
      <age>10</age>
      <link></link>
</student>
```

Lưu ý rằng phần tử liên kết là một phần của biểu diễn, nhưng đang trống vì giá trị này được tạo trong thời gian chạy và không được tạo bởi ứng dụng khách gửi yêu cầu POST. Đây chỉ là một quy ước cho ví dụ này; tuy nhiên, ứng dụng khách sử dụng dịch vụ web có thể chỉ định cấu trúc của URI.

Sơ đồ trình tự của yêu cầu POST sẽ như sau:



Hình 5-2. Sơ đồ tương tác sử dụng phương thức POST

Trình tự cụ thể như sau:

1. Ứng dụng khách Java đưa ra yêu cầu đối tới URI http://restfuljava.com/students/Jane, với phương thức HTTP POST.
2. Yêu cầu POST có tải trọng XML là biểu diễn của Jane.
3. Máy chủ nhận yêu cầu và để cho nền tảng REST xử lý; mã trong nền tảng thực thi các lệnh thích hợp để lưu trữ biểu diễn.
4. Sau khi hoàn tất việc lưu trữ tài nguyên mới, phản hồi sẽ được gửi lại: nếu thành công, mã 200 sẽ được gửi; nếu lỗi, cũng sẽ có mã thích hợp tương ứng.

5.2.4.3 Phương thức PUT – Cập nhập tài nguyên

Phương thức PUT được sử dụng để CẬP NHẬT tài nguyên.

Để cập nhật một tài nguyên, trước tiên máy khách cần có biểu diễn của tài nguyên; sau đó, tại máy khách, tài nguyên được cập nhật với (các) giá trị mới; và cuối cùng, tài nguyên được cập nhật thông qua yêu cầu PUT với tải trọng là biểu diễn mới của tài nguyên.

Giả sử máy khách, thông qua yêu cầu GET đã có biểu diễn và muốn thay đổi tuổi của Jave từ 10 thành 12. Biểu diễn ban đầu của Jane như sau:

```
<student>
    <name>Jane</name>
```

<age>10</age>

<link>/students/Jane</link>

</student>

Thay đổi tuổi của Jane thành 12, biểu diễn mới sẽ như sau:

<student>

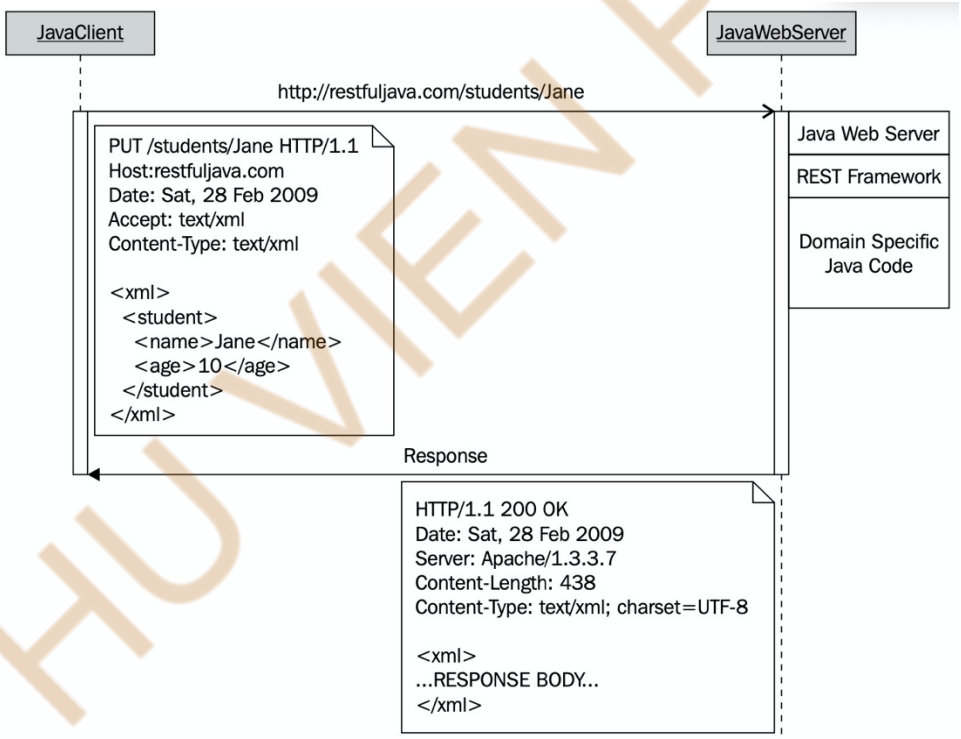
<name>Jane</name>

<age>12</age>

<link>/students/Jane</link>

</student>

Đến bước này thì dịch vụ web đã sẵn sàng kết nối để cập nhật Jane bằng cách gửi yêu cầu PUT tới http://restfuljava.com/students/Jane. Biểu đồ trình tự của yêu cầu PUT được thể hiện như sau:



Hình 5-3. Sơ đồ tương tác sử dụng phương thức PUT

Trình tự chi tiết cụ thể:

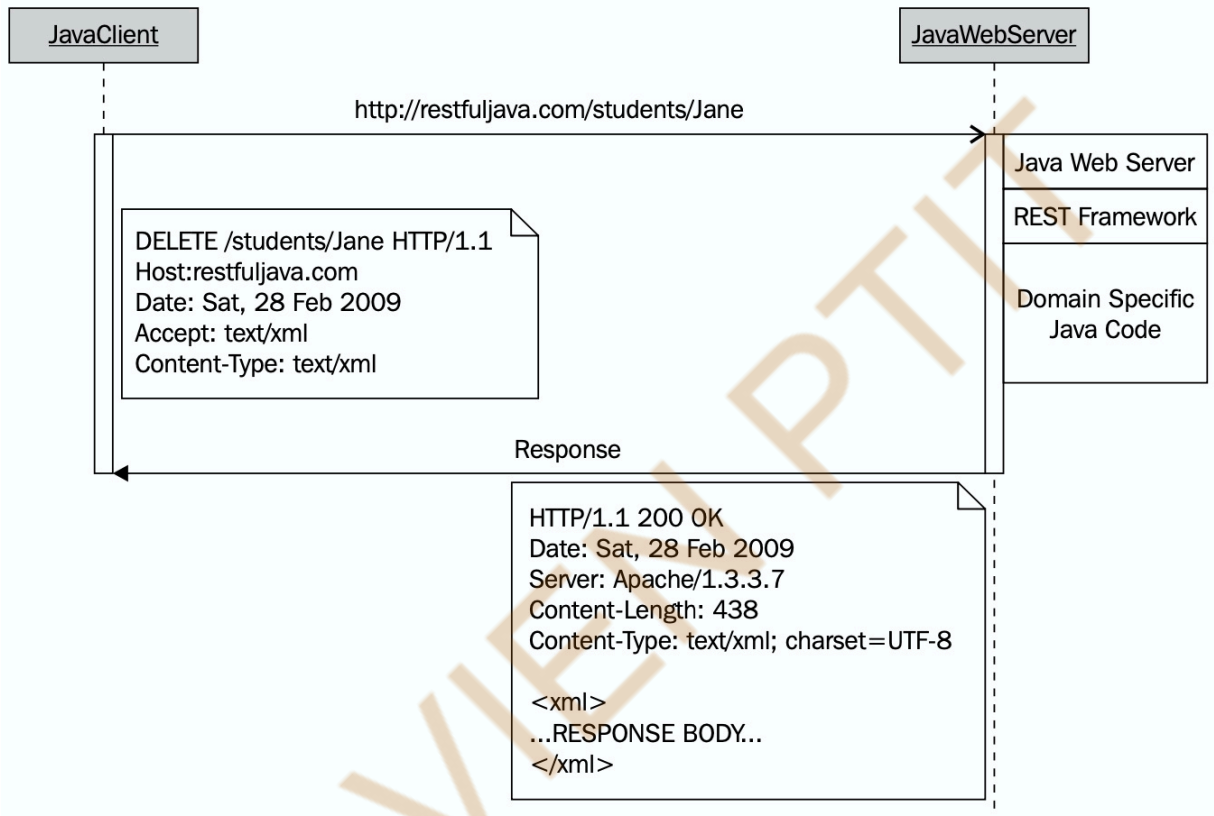
1. Ứng dụng khách Java gửi yêu cầu PUT tới http://restfuljava.com/students/Jane, với tải trọng là XML mới.
2. Máy chủ nhận yêu cầu và để nền tảng REST xử lý. Tại thời điểm này, mã nguồn sẽ thực thi các lệnh thích hợp để cập nhật biểu diễn của Jane.
3. Sau khi hoàn thành, phản hồi sẽ được gửi lại.

5.2.4.4 Phương thức DELETE – Xóa tài nguyên

Phương thức DELETE được sử dụng để XÓA các biểu diễn.

Việc xóa một tài nguyên sẽ sử dụng cùng một URI mà ta đã sử dụng ở trên.

Giả sử để xóa Jane khỏi dữ liệu. Một yêu cầu DELETE sẽ được gửi tới dịch vụ qua URI http://restfuljava.com/students/Jane. Biểu đồ trình tự cho yêu cầu DELETE như sau:



Hình 5-4. Sơ đồ tương tác sử dụng phương thức DELETE

Trình tự thực thi cụ thể như sau:

1. Ứng dụng khách Java gửi yêu cầu DELETE tới http://restfuljava.com/students/Jane.

2. Máy chủ nhận yêu cầu và chuyển nền tảng REST xử lý. Tiếp đến, mã nguồn sẽ thực hiện lệnh thích hợp để xóa biểu diễn của Jane.

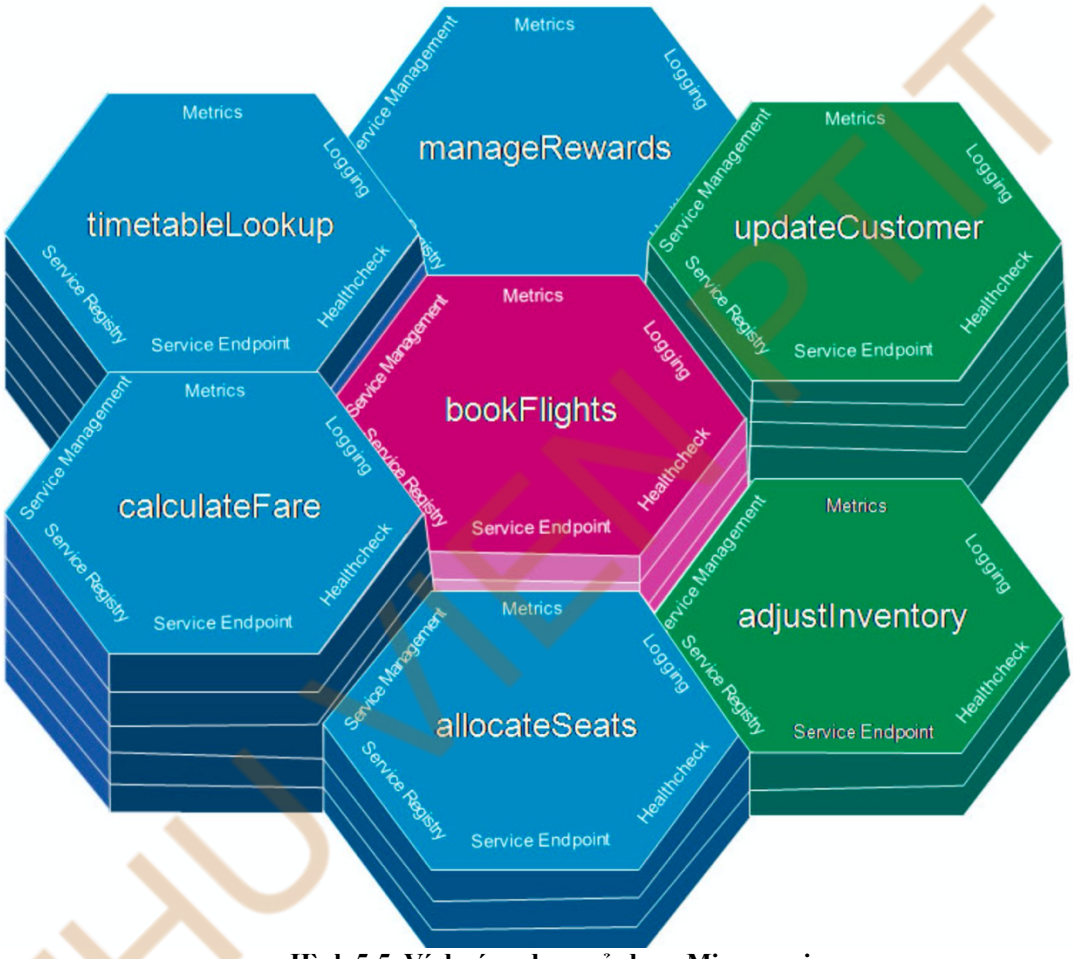
3. Sau khi hoàn thành, một phản hồi sẽ được gửi lại.

Phần này đã trình bày các hành động chính có thể thực hiện được với tài nguyên qua dịch vụ web RESTful. Thông tin khách hàng đang làm gì với biểu diễn tài nguyên, cách dịch vụ web triển khai lưu trữ dữ liệu và công nghệ nào được sử dụng cho dịch vụ không được đề cập. Tuy nhiên, để một dịch vụ web hoạt động tốt, nó chỉ cần tuân thủ các nguyên tắc RESTful: máy khách và máy chủ giao tiếp qua HTTP, sử dụng giao thức truyền thông để gửi các yêu cầu và biểu diễn tài nguyên được gửi qua lại thông qua các URI không thay đổi..

5.3 KIẾN TRÚC VI DỊCH VỤ

5.3.1 Khái niệm

Vi dịch vụ - Microservices là một phong cách kiến trúc, trong đó các ứng dụng phần mềm lớn, phức tạp được tạo từ một hoặc nhiều dịch vụ. Microservice có thể được triển khai độc lập với nhau và được ghép nối lỏng. Mỗi microservices chỉ tập trung vào việc hoàn thành một nhiệm vụ và thực hiện tốt một nhiệm vụ đó. Nhiệm vụ này thể hiện một nghiệp vụ nhỏ cụ thể. Hình dưới thể hiện ứng dụng mẫu sử dụng microservices



Hình 5-5. Ví dụ ứng dụng sử dụng Microservices

Ngoài ra, microservices có thể được phát triển bằng bất kỳ ngôn ngữ lập trình nào. Chúng giao tiếp với nhau bằng các giao diện lập trình ứng dụng trung lập về ngôn ngữ (API) như REST. Microservices cũng có ngữ cảnh giới hạn: nó không cần biết phải biết kiến trúc, hoặc việc thực thi của các microservice khác.

Các phần tiếp theo sẽ phân tích kỹ hơn về khai niệm của microservice.

5.3.3.1 Nhỏ và tập trung

Microservices cần tập trung vào một đơn vị công việc, do đó chúng có quy mô nhỏ. Không có quy tắc nào về mức độ nhỏ của một microservice. Một hướng dẫn thường được tham khảo là quy tắc Nhóm Two-Pizza, quy tắc này nêu rõ nếu không thể cung cấp cho nhóm xây dựng một microservice với hai chiếc pizza, thì microservice quá lớn. Tất nhiên, nếu muốn ta có thể làm

cho microservice đủ nhỏ để có thể viết lại và duy trì toàn bộ microservice dễ dàng trong một nhóm nếu cần.

Một microservice cũng cần được coi như một ứng dụng hoặc một sản phẩm. Nó nên có kho quản lý mã nguồn riêng và đường dẫn phân phối riêng để xây dựng và triển khai. Mặc dù chủ sở hữu sản phẩm có thể ủng hộ việc tái sử dụng microservice, nhưng việc sử dụng lại không phải là động lực kinh doanh duy nhất cho microservice. Những vấn đề khác, chẳng hạn như tối ưu hóa vị trí để cải thiện khả năng phản hồi của giao diện người dùng (UI) và để có thể đáp ứng nhu cầu của khách hàng nhanh hơn.

Mức độ chi tiết của microservice cũng có thể được xác định dựa trên nhu cầu nghiệp vụ. Theo dõi gói hàng, dịch vụ báo giá bảo hiểm xe hơi và dự báo thời tiết là những ví dụ về dịch vụ do các nhà cung cấp dịch vụ bên thứ ba khác cung cấp hoặc được cung cấp như một dịch vụ tính phí lõi.

5.3.3.2 Ghép nối lỏng

Ghép nối lỏng là một đặc điểm cơ bản của microservices, để cho phép người dùng có thể tự triển khai một microservice đơn lẻ. Phối hợp với các microservice khác cho việc triển khai là không bắt buộc. Ghép nối lỏng cho phép việc triển khai trở thành thường xuyên và nhanh chóng, do đó đáp ứng được các tính năng và khả năng cần thiết cho người tiêu dùng.

5.3.3.3 Trung lập với ngôn ngữ

Sử dụng đúng công cụ cho đúng công việc có vai trò quan trọng trong việc thành công của công việc. Các microservice cần được xây dựng bằng ngôn ngữ lập trình và công nghệ phù hợp nhất cho một nhiệm vụ cụ thể. Các microservice được cấu thành cùng nhau để tạo thành một ứng dụng phức tạp, chúng không cần phải được viết bằng cùng một ngôn ngữ lập trình. Tùy vào trường hợp cụ thể, Java có thể là ngôn ngữ được sử dụng hoặc có thể là Python hoặc ngôn ngữ khác.

Giao tiếp với microservices thông qua các API trung lập về ngôn ngữ, thường là API HTTP, ví dụ REST. Nên chuẩn hóa về sự tích hợp chứ không phải trên nền tảng mà microservice sử dụng. Ngôn ngữ trung lập cho phép sử dụng các kỹ năng hiện có hoặc ngôn ngữ tối ưu nhất.

5.3.3.2 Ngữ cảnh giới hạn

Ý của của ngữ cảnh giới hạn là một microservice cụ thể không "biết" bất cứ điều gì về việc triển khai cơ bản của các microservices khác. Nếu vì bất kỳ lý do gì mà một microservice cần biết bất cứ điều gì về một microservice khác (ví dụ, nó hoạt động gì hoặc nó cần được gọi như thế nào), thì sẽ không còn ngữ cảnh bị ràng buộc.

5.3.3.2 Kiến trúc Microservices và Nguyên khối

Bảng 5-1 so sánh microservices và kiến trúc nguyên khối.

Loại	Kiến trúc nguyên khối	Kiến trúc microservices
Mã	Một mã cơ sở duy nhất cho toàn bộ ứng dụng.	Nhiều mã cơ sở. Mỗi microservice có cơ sở mã riêng.

Tính dễ hiểu	Thường khó hiểu và khó bảo trì.	Khả năng đọc tốt hơn và dễ bảo trì hơn.
Triển khai	Triển khai phức tạp với các khoảng bảo trì và thời gian ngừng hoạt động theo lịch trình.	Triển khai đơn giản vì mỗi microservice có thể được triển khai riêng lẻ, với thời gian chết tối thiểu gần như bằng không.
Ngôn ngữ	Thường được phát triển hoàn toàn bằng một ngôn ngữ lập trình.	Mỗi microservice có thể được phát triển bằng một ngôn ngữ lập trình khác nhau.
Mở rộng quy mô	Yêu cầu mở rộng quy mô toàn bộ ứng dụng ngay cả khi các nút cổ chai được bản địa hóa.	Cho phép mở rộng các dịch vụ cổ chai mà không cần mở rộng toàn bộ ứng dụng.

5.3.2 Đặc trưng

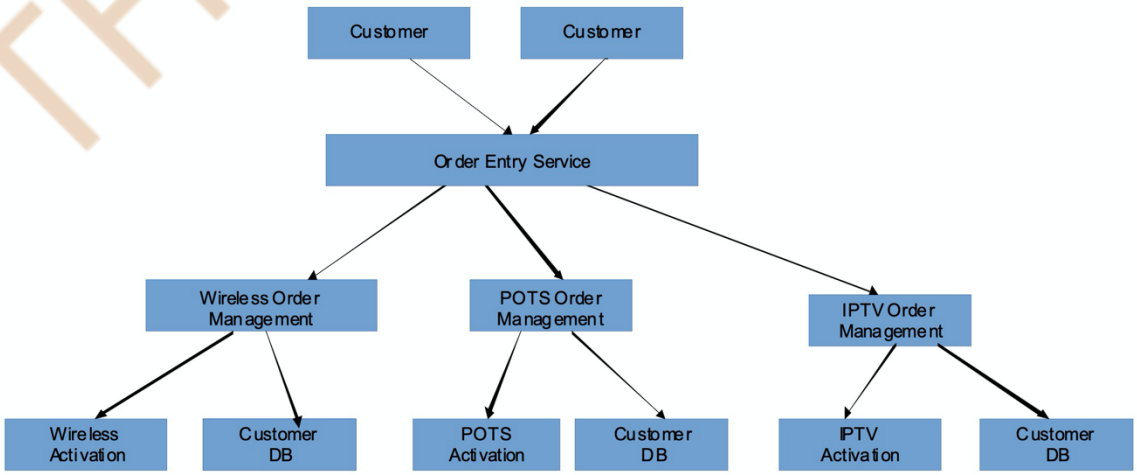
5.3.2.1 Hướng nghiệp vụ

Khi một hệ thống được chia và cấu thành từ các dịch vụ nhỏ, có một nguy cơ đó là việc phân rã này có thể được thực hiện dọc theo các ranh giới trong tổ chức, ví dụ chia thành các dịch vụ riêng theo phòng, ban, tổ ... Điều này dẫn tới hệ thống có thể mong manh hơn, vì có nhiều phần độc lập cần quản lý, đồng thời cũng có phần tích hợp không được tốt (mặc dù vẫn có thể tương tác được với các dịch vụ khác).

Ngoài ra, nếu các dịch vụ tạo nên hệ thống không được thiết kế hướng tới mục tiêu một cách chính xác, chúng không thể được sử dụng lại. Chi phí phát triển và bảo trì cũng có thể tăng nếu các dịch vụ được thiết kế dọc theo ranh giới tổ chức hoặc công nghệ.

Điều quan trọng là phải thiết kế microservices cần hướng tới mục tiêu nghiệp vụ. Việc này đòi hỏi các nhóm ở trong tổ chức cùng ngồi lại với nhau để thiết kế các dịch vụ, thay vì sử dụng microservices để định tuyến cuộc gọi giữa các nhóm.

Hình 5-5 mô tả kịch bản thiết kế dịch vụ theo phong cách "cũ".

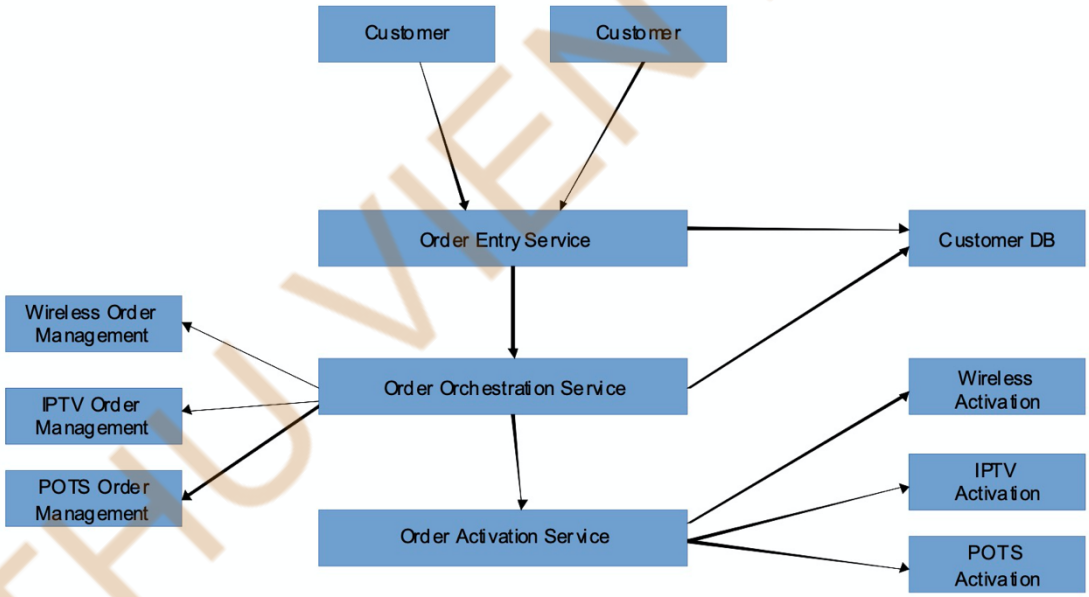


Hình 5-5. Thiết kế dịch vụ phong cách cũ

Trong kịch bản trước, việc thiết kế hệ thống bắt chước và giựa trên cấu trúc giao tiếp của tổ chức, dẫn đến một số sai sót được đưa vào trong các dịch v đang ra phải hoàn toàn biệt lập. Hệ thống đặt hàng không thể mở rộng quy mô, bởi vì mỗi tổ chức có hệ thống kích hoạt và quản lý đơn hàng riêng, mỗi tổ chức có quan điểm riêng về khách hàng và không có hai bộ phận nào trong tổ chức có cùng quan điểm về khách hàng.

Các dịch vụ được thiết kế trong kịch bản trước cũng sẽ bị ảnh hưởng bởi những thay đổi thường ảnh hưởng đến tổ chức, chẳng hạn như ngừng hoạt động, sáp nhập & mua lại cũng như các thay đổi đối với thiết bị mạng được sử dụng để cung cấp dịch vụ. Sẽ không vô lý nếu kỳ vọng rằng các dịch vụ như vậy đang được từng bộ phận đo lường, với các thước đo lỗi, lỗi và hiệu suất riêng. Tuy nhiên, hệ thống nhập lệnh nói chung không thể phục vụ mục đích của tổ chức.

Cách tiếp cận microservice hoạt động hiệu quả nếu các nhà thiết kế và triển khai của từng bộ phận trong tổ chức giao tiếp với nhau để cùng nhau thiết kế các dịch vụ nhằm phục vụ một mục đích kinh doanh chung. Hình 5-6 cho thấy ứng dụng được thiết kế lại với cách tiếp cận microservices. Lưu ý rằng cấu trúc tương tự cũng sẽ là kết quả của việc sử dụng phương pháp SOA.



Hình 5-6 Ứng dụng được thiết kế lại với cách tiếp cận microservices

Trong kịch bản trước, các nhóm chức năng chéo đã cùng nhau thiết kế một dịch vụ nhập đơn hàng chung. Khi một đơn đặt hàng được nhận, nó được lưu trữ trong cơ sở dữ liệu, cung cấp một cái nhìn thống nhất duy nhất về khách hàng. Sau khi nhận được đơn đặt hàng, đơn đặt hàng sẽ được gửi đến dịch vụ điều phối, dịch vụ này gọi từng hệ thống đặt hàng riêng lẻ, theo yêu cầu. Các cuộc gọi này sau đó sẽ kích hoạt từng phần của đơn đặt hàng.

Các thiết kế này không bắt chước các ranh giới về tổ chức, công nghệ hoặc truyền thông. Do đó, các dịch vụ có thể chịu được những thay đổi đối với tổ chức (chẳng hạn như các sản phẩm hoặc dịch vụ mới được thêm vào hoặc mua lại mới) và những thay đổi về nhân sự. Ngoài ra, các dịch vụ được tách biệt để hỗ trợ một chức năng kinh doanh.

5.3.2.2 Chịu lỗi

Kỹ thuật phần mềm có thể học tập nhiều kiến thức từ kỹ thuật dân dụng, chẳng hạn kỹ thuật xây dựng, thiết kế đường xá, cầu, kênh, đập và các tòa nhà. Kỹ sư dân dụng thường thiết kế hệ thống bao gồm cả việc dự kiến sự cố của từng bộ phận riêng lẻ và xây dựng một số lớp dự phòng để đảm bảo các tòa nhà, công trình vẫn an toàn và ổn định dù có lỗi ở một số bộ phận.

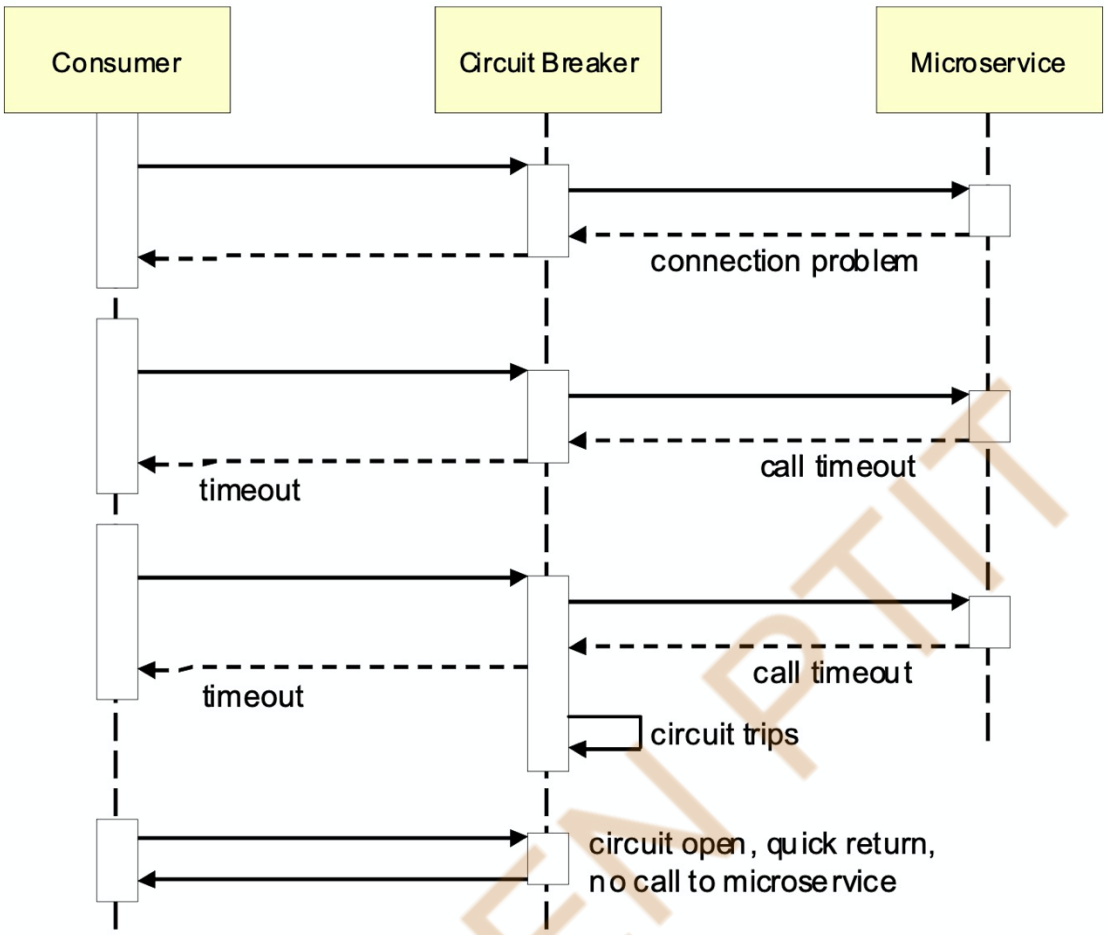
Tư duy tương tự có thể được áp dụng cho kỹ thuật phần mềm. Các lỗi độc lập là không thể tránh khỏi, nhưng mục tiêu thiết kế là giữ cho hệ thống hoạt động càng lâu càng tốt, cho dù có lỗi ở một số bộ phận. Các kỹ thuật, chẳng hạn như mô hình hóa lỗi và tiêm lỗi (injection), nên được đưa vào như một phần của quá trình phát hành liên tục để thiết kế các hệ thống đáng tin cậy hơn. Có một số mẫu thiết kế để thiết kế cho phép chịu lỗi và để hệ thống hoạt động ổn định như sau.

Kỹ thuật ngắt mạch

Mô hình ngắt mạch thường được sử dụng để đảm bảo rằng khi có sự cố, dịch vụ không thành công sẽ không ảnh hưởng đến toàn bộ hệ thống. Điều này sẽ xảy ra nếu số lượng cuộc gọi đến dịch vụ không thành công cao và đối với mỗi cuộc gọi, ta sẽ phải đợi một khoảng thời gian chờ xảy ra trước khi tiếp tục. Thực hiện cuộc gọi đến dịch vụ không thành công và chờ đợi sẽ sử dụng tài nguyên mà cuối cùng sẽ làm cho hệ thống tổng thể không ổn định.

Mô hình ngắt mạch hoạt động giống như cầu dao trong hệ thống điện gia đình. Các cuộc gọi đến một microservice được bao bọc trong một đối tượng ngắt mạch. Khi một dịch vụ bị lỗi, đối tượng ngắt mạch cho phép các cuộc gọi tiếp theo tới dịch vụ cho đến khi đạt đến một ngưỡng cụ thể của các lần thử không thành công. Tại thời điểm đó, bộ ngắt mạch cho các lời gọi đến dịch vụ và bất kỳ cuộc gọi tiếp theo, sẽ bị ngắt. Thiết lập này giúp tiết kiệm tài nguyên và duy trì sự ổn định chung của hệ thống.

Hình 5-6 biểu diễn sơ đồ trình tự cho mô hình ngắt mạch.



Hình 5-6. Sơ đồ trình tự mô hình ngắt mạch

Khi ngắt mạch, một logic dự phòng có thể được khởi động. Logic dự phòng thường có ít chức năng hoặc không xử lý gì, chỉ trả về một giá trị mặc định nào đó. Logic dự phòng cần được thiết kế để có ít khả năng bị lỗi.

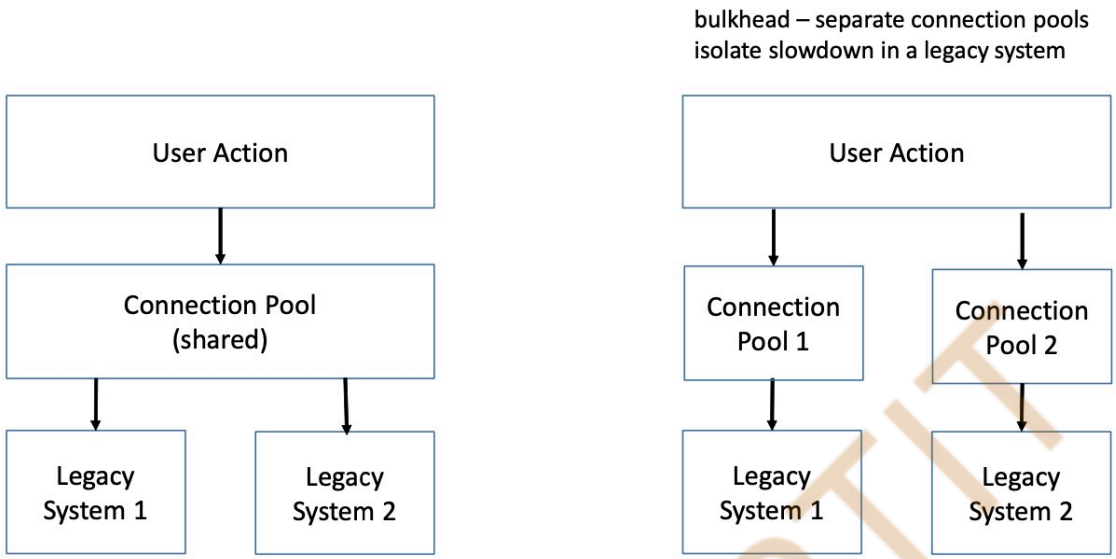
Vách ngăn

Lấy ví dụ về cấu tạo của vỏ tàu, chúng thường bao gồm một số vách ngăn kín nước riêng lẻ, để nếu một trong các vách ngăn bị hư hỏng, nước chỉ vào ở một số không gian nhỏ và bị chặn lại bởi các vách khác không bị lỗi.

Hướng tiếp cận phân vùng này cũng có thể được sử dụng trong thiết kế phần mềm, để cô lập sự cố đối qua các phần nhỏ của hệ thống. Ranh giới dịch vụ (chính bản thân microservice) đóng vai trò như một vách ngăn để cô lập bất kỳ lỗi nào nếu phát sinh. Việc chia nhỏ chức năng (như cách chúng ta cũng làm trong kiến trúc SOA) thành các microservices riêng biệt nhằm mục đích tách biệt tác động của lỗi trong một microservice.

Mẫu vách ngăn cũng có thể được áp dụng trong chính các microservices. Ví dụ, hãy xem xét một nhóm luồng đang được sử dụng để tiếp cận hai hệ thống hiện có. Nếu một trong các hệ thống hiện có bắt đầu gặp sự cố chậm và khiến nhóm luồng cạn kiệt, quyền truy cập vào hệ thống hiện có khác cũng sẽ bị ảnh hưởng. Việc có các nhóm luồng riêng biệt sẽ đảm bảo rằng sự chậm lại trong một hệ thống hiện có sẽ chỉ làm cạn kiệt nhóm luồng của chính nó và không ảnh hưởng đến quyền truy cập vào hệ thống hiện có khác.

Xem Hình 5-6 minh họa về cách sử dụng nhóm luồng riêng biệt này.



Hình 1-6. Ví dụ sử dụng mẫu vách ngăn: Sử dụng các nhóm luồng riêng biệt để cô lập lỗi

5.3.2.3 Quản lý dữ liệu phi tập trung

Trong ứng dụng nguyên khối, ta có thể dễ dàng xử lý các giao dịch vì tất cả thành phần đều là một phần của nguyên khối. Khi chuyển sang kiến trúc phân tán microservices, ta cần phải quan tâm và có khả năng đối phó với các giao dịch trải rộng trên nhiều dịch vụ. Trong kiến trúc microservices, ưu tiên dành cho BASE (Basically Available, Soft state, Eventual consistency - Cơ bản có sẵn, Trạng thái mềm, Tính nhất quán cuối cùng) hơn ACID (Atomicity, Consistency, Isolation, Durability - Tính nguyên tử, Tính nhất quán, Tính cách ly, Độ bền). Nên tránh các giao dịch phân tán bất cứ khi nào có thể.

Lý tưởng nhất là mọi microservice đều quản lý cơ sở dữ liệu của riêng mình. Điều này đảm bảo tính bền bỉ, sử dụng các cơ sở dữ liệu khác nhau (ví dụ: Cloudant so với MongoDB, cả hai đều là NoSQL) và các kiểu lưu trữ dữ liệu khác nhau (chẳng hạn như SQL, NoSQL, đồ thị). Tuy nhiên, chúng ta có thể cần phải có nhiều microservice sử dụng cùng một cơ sở dữ liệu vì một trong nhiều lý do, ví dụ: để bảo toàn bản chất ACID của một giao dịch mà nếu không sẽ được phân phối trên các microservice và cơ sở dữ liệu.

Cho dù lý do là gì, cần phải suy nghĩ cẩn thận khi chia sẻ cơ sở dữ liệu trên các microservices. Những ưu và khuyết điểm của việc làm như vậy phải được xem xét. Chia sẻ cơ sở dữ liệu vi phạm một số nguyên tắc của kiến trúc dựa trên microservices. Ví dụ, ngữ cảnh không còn bị ràng buộc, nơi hai dịch vụ chia sẻ cơ sở dữ liệu cần biết về nhau và những thay đổi trong cơ sở dữ liệu dùng chung cần được phối hợp giữa hai dịch vụ.

Nói chung, mức độ chia sẻ giữa các microservices nên được hạn chế hết mức có thể để làm cho các microservices được liên kết lỏng lẻo nhất có thể.

5.3.2.4 Khả năng có thể khám phá

Như ta đã mô tả trước đó, kiến trúc microservices yêu cầu tạo ra các dịch vụ đáng tin cậy và có khả năng chịu lỗi. Có nghĩa là, khi cơ sở hạ tầng bên dưới được tạo ra hoặc hủy, các

microservice vẫn có thể được cấu hình mơi/lại với vị trí của các dịch vụ khác mà chúng cần kết nối.

Với việc sử dụng điện toán đám mây và bộ chứa để triển khai microservices, các dịch vụ này cần được cấu hình lại động. Khi phiên bản dịch vụ mới được tạo, phần còn lại của mạng có thể nhanh chóng tìm thấy và bắt đầu giao tiếp dịch vụ đó.

Hầu hết các mô hình khám phá dịch vụ dựa vào dịch vụ đăng ký mà tất cả các dịch vụ đều đăng ký rõ ràng và thực hiện các cuộc gọi đến nó sau đó để liên lạc với các dịch vụ khác. Hiện có một số dịch vụ đăng ký mã nguồn mở cung cấp khả năng khám phá dịch vụ, chẳng hạn như Zookeeper, Consul và Eureka.

Tuy nhiên không phải tất cả các mô hình khám phá dịch vụ đều dựa vào dịch vụ đăng ký. Ví dụ: Cloud Foundry, nền tảng mã nguồn mở như một dịch vụ (PaaS) mà Bluemix được xây dựng trên đó, không dựa vào dịch vụ đăng ký. Các dịch vụ trong Cloud Foundry được quảng bá tới Bộ điều khiển đám mây để quảng bá về sự tồn tại của dịch vụ và tính khả dụng của nó.

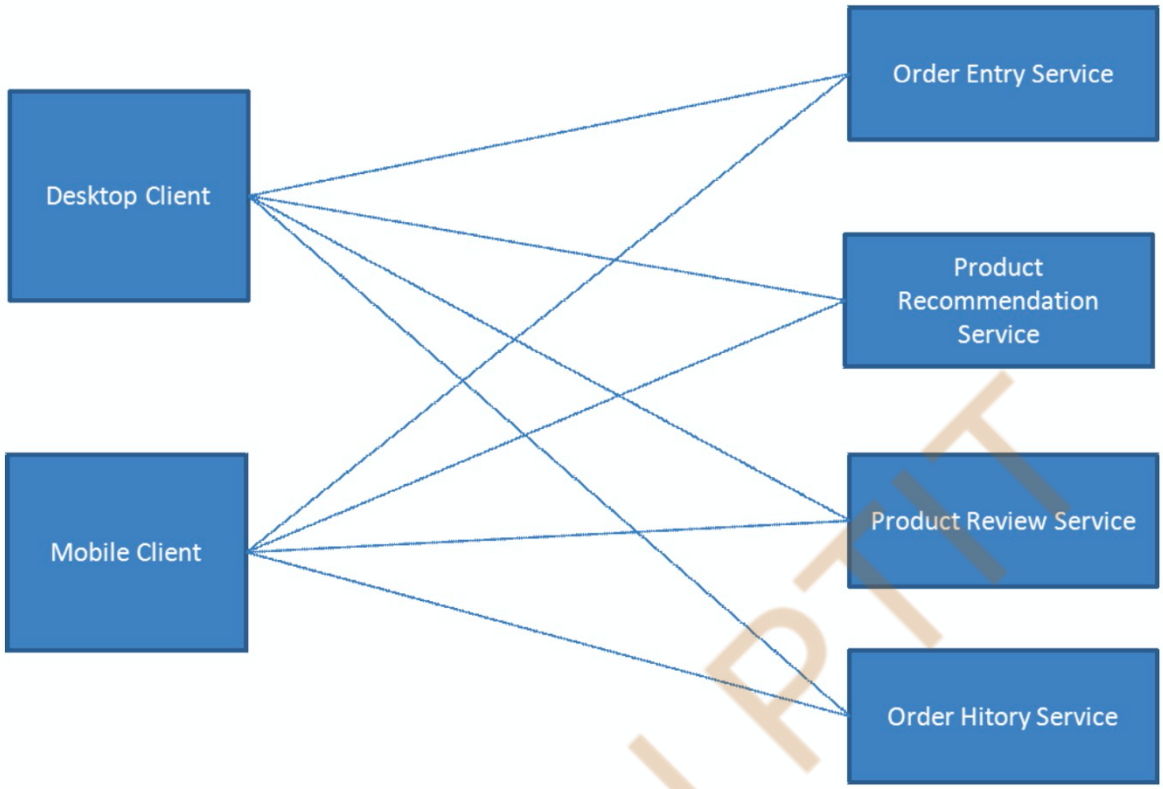
Dịch vụ khám phá tốt là một nguồn siêu dữ liệu phong phú mà các dịch vụ đăng ký có thể cung cấp và sau đó có thể được sử dụng bằng cách yêu cầu các dịch vụ đưa ra quyết định phù hợp về cách sử dụng nó. Ví dụ: yêu cầu (hoặc khách hàng) dịch vụ có thể hỏi về việc liệu một dịch vụ có bất kỳ phụ thuộc cụ thể nào không, hàm ý của chúng và liệu nó có thể đáp ứng bất kỳ yêu cầu cụ thể nào hay không. Một số dịch vụ khám phá cũng có thể có khả năng cân bằng tải.

5.3.2.5 Thiết kế giao tiếp liên dịch vụ

Khi các microservice được trải rộng trên bộ chứa triển khai (máy chủ, máy ảo (VM) hoặc các bộ chứa khác), trong nhiều trường hợp chúng cần giao tiếp với nhau. Đối với giao tiếp một chiều, có thể chỉ cần sử dụng hàng đợi tin nhắn, tuy nhiên với giao tiếp 2 chiều đồng bộ thì lại khác.

Trong các ứng dụng nguyên khối, máy khách của ứng dụng, chẳng hạn như trình duyệt và ứng dụng gốc, gửi yêu cầu HTTP tới một bộ cân bằng tải, cho phép lan truyền các yêu cầu đến một số phiên bản giống hệt nhau của ứng dụng. Nhưng trong kiến trúc microservice, mô hình nguyên khối đã được thay thế bằng một tập hợp các dịch vụ.

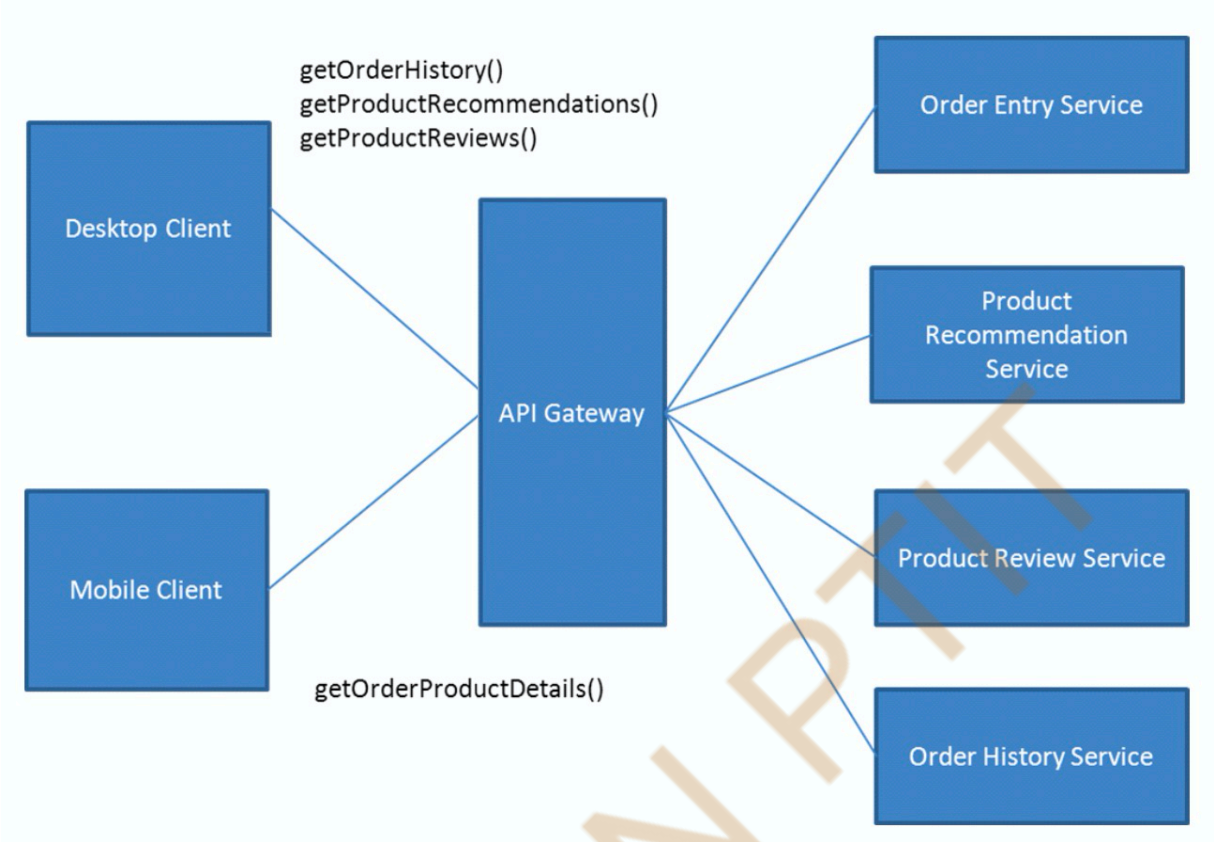
Một kịch bản có thể là các máy khách của hệ thống có thể thực hiện các lời gọi qua API RESTful, như được mô tả trong Hình 5-7.



Hình 5-7. Gọi API RESTful giữa máy khách và dịch vụ

Về thiết kế kịch bản các lời gọi rất hoàn hảo, tuy nhiên, đi vào chi tiết, nhiều lúc, các dịch vụ không khớp với thiết kế. Một số máy khách chỉ đơn thuần là các lời gọi dưới dạng "trò chuyện" một – một; những có nhiều máy khác có thể cần yêu cầu nhiều lời gọi khác để có được dữ liệu cần thiết. Bởi vì các dịch vụ có thể có quy mô khác nhau, nên có những thách thức khác cần được giải quyết, như xử lý các lỗi một phần.

Một cách tiếp cận tốt hơn là các máy khách thực hiện một vài yêu cầu, hoặc chỉ một, bằng cách sử dụng giao diện người dùng như API Gateway, như trong Hình 5-8.



Hình 5-8. Giao tiếp microservice thông qua API Gateway

API Gateway nằm giữa máy khách và microservices, đồng thời cung cấp các API được thiết kế riêng cho khách hàng. API Gateway chủ yếu để che giấu sự phức tạp về công nghệ (ví dụ: kết nối với máy tính lớn) so với sự phức tạp của giao diện. Một cách không chính thức, API Gateway có thể xem như mặt tiền (facade) của dịch vụ. Mặt tiền cung cấp một cái nhìn thống nhất về phần bên trong phức tạp cho các máy khách bên ngoài, và API Gateway cũng cung cấp một cái nhìn thống nhất về các tài nguyên bên ngoài cho phần bên trong của một ứng dụng.

Giống như mẫu thiết kế mặt tiền (facade design pattern), API Gateway cung cấp giao diện đơn giản hóa cho máy khách, làm cho các dịch vụ dễ sử dụng, dễ hiểu và dễ kiểm tra hơn. Vì nó có thể cung cấp các mức độ chi tiết khác nhau cho máy tính và trình duyệt. API Gateway có thể cung cấp các API chi tiết cho cả máy khách trên thiết bị di động và cả API chi tiết cho các ứng dụng máy tính để bàn có thể sử dụng mạng hiệu suất cao.

Trong trường hợp này, API Gateway giảm bớt sự "trò chuyện" đơn thuần bằng cách cho phép khách hàng thu gọn nhiều yêu cầu thành một yêu cầu duy nhất được tối ưu hóa cho một ứng dụng khách nhất định (chẳng hạn như ứng dụng khách di động). Lợi ích là thiết bị sau đó sẽ chỉ bị độ trễ mạng một lần và tận dụng kết nối có độ trễ thấp và phía máy chủ phần cứng mạnh mẽ hơn.

Các phần sau mô tả một số cân nhắc khác để thực hiện liên lạc giữa các dịch vụ

HTTP đồng bộ và Thông điệp không đồng bộ

Có hai cách tiếp cận chính để giao tiếp giữa các quá trình:

- Cơ chế dựa trên HTTP đồng bộ, ví dụ như REST, SOAP hoặc WebSockets, cho phép giữ kênh giao tiếp giữa trình duyệt và máy chủ mở cho yêu cầu / phản hồi theo hướng sự kiện
- Nhắn tin không đồng bộ thông qua trình môi giới tin nhắn

Liên lạc đồng bộ là phương pháp đơn giản và phổ biến, thân thiện với tường lửa. Điểm bất lợi là nó không hỗ trợ các mô hình khác, chẳng hạn như mô hình xuất bản/đăng ký. Kiểu nhắn tin xuất bản/đăng ký linh hoạt hơn kiểu xếp hàng điểm-điểm, vì nó cho phép nhiều người nhận đăng ký luồng tin nhắn. Điều này cho phép dễ dàng thêm nhiều microservices vào ứng dụng.

Nó cũng không cho phép xếp hàng các yêu cầu, hoạt động như một loại giảm tải yêu cầu. Liên lạc không đồng bộ tách các dịch vụ khỏi nhau theo thời gian, cho phép chuỗi gửi tiếp tục hoạt động trong khi hệ thống nhắn tin chịu trách nhiệm về tin nhắn, giữ nó an toàn cho đến khi nó có thể được gửi đến đích.

Với các ứng dụng đồng bộ, cả máy khách và máy chủ phải đồng thời khả dụng, máy khách luôn cần biết máy chủ và cổng của máy chủ, điều này không phải lúc nào cũng dễ hiểu khi các dịch vụ tự động mở rộng quy mô trong triển khai trên đám mây. Trong các tình huống như vậy, một trong các cơ chế khám phá dịch vụ được mô tả trước đây là bắt buộc.

Ngoài ra, cơ chế không đồng bộ sử dụng một trình môi giới thông điệp, phân tách thông điệp của bên tiêu dùng và bên tạo ra. Môi giới thông điệp thực hiện việc lưu đệm các tin nhắn cho đến khi bên tiêu dùng có thể xử lý chúng.

Một lợi ích khác của thông điệp là các ứng dụng back-end đằng sau hàng đợi có thể phân phối khối lượng công việc giữa một số máy khách. Loại kịch bản giảm tải này cho phép các nhà phát triển xây dựng các ứng dụng đáp ứng, có thể mở rộng được. Việc triển khai một hoặc nhiều trường hợp của ứng dụng back-end cho phép ứng dụng đưa vào tiếp tục công việc của nó mà không cần đợi ứng dụng back-end hoàn tất.

Có những ưu và nhược điểm cho cả hai cách tiếp cận. Hầu hết các ứng dụng sử dụng kết hợp hai cách tiếp cận để phù hợp với nhu cầu của họ.

5.3.2.6 Sự phức tạp

Một kiến trúc microservices dẫn tới việc sử dụng nhiều tài nguyên các hoạt động: có một sự gia tăng lớn trong di chuyển các thành phần, và do đớ dẫn tới sự phức tạp trong hoạt động của hệ thống. Hợp lý hóa hành vi của từng thành phần riêng lẻ có thể đơn giản hơn, nhưng hiểu được hành vi của toàn bộ hệ thống thì khó hơn nhiều.

Việc điều chỉnh một hệ thống dựa trên microservices, có thể bao gồm hàng chục quy trình, bộ cân bằng tải và lớp nhắn tin, đòi hỏi phải có cơ sở hạ tầng hoạt động và giám sát chất lượng cao.

Danh sách sau đây mô tả một số cân nhắc để đối phó với sự phức tạp do sử dụng microservices:

Kiểm thử

Việc kiểm thử một hệ thống phức tạp yêu cầu các bộ kiểm thử đơn vị và tích hợp, và một khung kiểm tra mô phỏng các thành phần bị lỗi hoặc không thành công, sự xuống cấp của dịch vụ, kiểm tra chịu lỗi và hành vi tổng thể trong suốt một ngày. Các khuôn khổ kiểm thử như vậy đảm bảo tính đúng đắn của hệ thống và khả năng phục hồi tổng thể, do đó có thể được sử dụng như một công cụ giám sát.

Do tập trung vào việc đảm bảo rằng hệ thống hoạt động tốt khi một thành phần bị lỗi, nên việc kiểm tra lỗi trong môi trường sản xuất trong kiến trúc microservices không phải là hiếm. Các thử nghiệm tự động buộc một thành phần không thành công cho phép kiểm tra khả năng phục hồi của hệ thống và kiểm tra khả năng giám sát.

Ngoài các hệ thống giám sát thông thường, các dịch vụ được thiết kế tốt có thể cung cấp thêm thông tin kiểm tra bằng cách sử dụng xuất bản/đăng ký qua hàng đợi nhắn tin. Bằng cách này, dịch vụ giám sát có thể thu thập thông tin, cung cấp các chỉ số cấu hình và sức khỏe của hệ thống.

Các chỉ số này có thể được sử dụng để đảm bảo rằng dịch vụ có đủ năng lực để phục vụ khối lượng yêu cầu hiện tại và so sánh hiệu suất hiện tại với kỳ vọng. Kiểm tra sức khỏe hệ thống tùy chỉnh có thể được đăng ký để thu thập dữ liệu bổ sung. Tất cả thông tin thu thập sau đó có thể được tổng hợp và hiển thị trong bảng điều khiển giám sát.

Thách thức hoạt động của việc duy trì và chạy các microservice yêu cầu các kỹ năng DevOps chất lượng cao được nhúng trong nhóm phát triển. Nhóm cần tập trung vào hoạt động và nhận thức về sản xuất.

Giám sát

DevOps

Lập trình đa thức

Việc sử dụng một cách ngẫu nhiên các microservices cũng có nghĩa là hệ thống thu được sẽ là đa phức, do đó dẫn tới thách thức trong việc duy trì một hệ thống siêu đa. Nên, quản trị viên cơ sở dữ liệu (DBA) hiện cần được thay thế bằng các lập trình viên có thể triển khai, chạy và tối ưu hóa các sản phẩm noSQL.

Việc duy trì hệ thống của các nhà phát triển, và nhà nghiên cứu có kỹ năng cũng quan trọng như việc duy trì hệ thống DevOps để đảm bảo rằng có đủ các nhà phát triển có kỹ năng để duy trì nền tảng và cải thiện thiết kế hệ thống khi có những tiến bộ mới. Lưu ý rằng hầu hết các tổ chức đều có một danh sách ngắn các ngôn ngữ lập trình mà họ sử dụng. Có những lợi ích cho điều đó từ quan điểm sử dụng lại mã và kỹ năng.

5.3.2.7 Sự tiến hóa trong thiết kế

Kiến trúc Microservices cho phép tiếp cận hướng thiết kế tiến hóa: có thể đưa các tính năng và khả năng mới vào ứng dụng bằng cách thay đổi một microservice. Việc này có thể thực hiện được thông qua việc triển khai và phát hành các microservice đã thay đổi. Thay đổi chỉ ảnh hưởng đến bên tiêu dùng của microservice và nếu giao diện không thay đổi, tất cả người tiêu dùng tiếp tục hoạt động bình thường.

Bởi vì các microservice được ghép nối lỏng, nhỏ, tập trung, và có bối cảnh giới hạn, các thay đổi đối với chúng có thể được thực hiện nhanh chóng và thường xuyên với rủi ro tối thiểu. Khả năng nâng cấp dễ dàng này của microservice cho phép việc tiến hóa, cập nhập trong thiết kế.

Bởi vì một microservice nhỏ, thường được thiết kế, phát triển và duy trì bởi một nhóm nhỏ, nên thông thường microservice sẽ được viết lại hoàn toàn, thay vì được nâng cấp hoặc bảo trì. Càng có nhiều microservice trong quá trình phân tách một ứng dụng nguyên khối, thì mỗi dịch vụ càng nhỏ; Mỗi cái càng nhỏ thì rủi ro trong việc triển khai / phát hành các thay đổi càng thấp.

Tuy nhiên, nếu microservice càng nhỏ thì có nhiều khả năng bản cập nhật ứng dụng sẽ yêu cầu triển khai nhiều microservice hơn, do đó làm tăng rủi ro. Các nguyên tắc của microservices, ví dụ khả năng triển khai độc lập và ghép nối lỏng, cần được xem xét khi xác định khả năng nào trở thành microservice.

5.3.3 Kiến trúc vi dịch vụ và Kiến trúc hướng dịch vụ

Phần này so sánh giữa cách tiếp cận microservices với kiến trúc hướng dịch vụ (SOA).

Sự so sánh là phức tạp và hơi không công bằng, bởi vì những người ủng hộ kiến trúc microservice không bao giờ khẳng định rằng nó đại diện cho một cách tiếp cận thực sự mới để xây dựng hệ thống phân tán. Vì vậy, so sánh giữa SOA và microservices thường được đề xuất bởi những người đề xuất SOA, những người muốn chứng minh rằng sự khác biệt như vậy không có trong thực tế.

Trước khi trả lời câu hỏi trực tiếp, như đã biết SOA là:

- Tập hợp các dịch vụ và chức năng phù hợp với nghiệp vụ mà doanh nghiệp muốn cung cấp cho khách hàng, đối tác hoặc các khu vực khác của tổ chức.
- Một phong cách kiến trúc yêu cầu nhà cung cấp dịch vụ, người yêu cầu dịch vụ có bản mô tả dịch vụ và có thể cả trung gian.
- Một tập hợp các nguyên tắc, mẫu và tiêu chí kiến trúc đề cập đến các đặc điểm, ví dụ như tính mô đun, tính đóng gói, ghép nối lỏng, tách biệt các mối quan tâm, tái sử dụng và khả năng kết hợp.
- Một mô hình lập trình hoàn chỉnh với các tiêu chuẩn, công cụ và công nghệ hỗ trợ các dịch vụ web, dịch vụ REST hoặc các loại dịch vụ khác.
- Một giải pháp phần mềm trung gian được tối ưu hóa cho việc lắp ráp, điều phối, giám sát và quản lý dịch vụ.

Với định nghĩa này, rõ ràng SOA có một tập hợp các mục tiêu và vấn đề toàn diện, sâu rộng đang hướng tới để giải quyết.

Mặc dù trong cả hai trường hợp đều đang nói về tập hợp các dịch vụ, nhưng tham vọng của các dịch vụ này là khác nhau. SOA cố gắng cung cấp dịch vụ cho bất kỳ ai muốn sử dụng chúng. Thay vào đó, microservices được tạo ra với mục tiêu tập trung và hạn chế hơn nhiều, nó hoạt động như một phần của hệ thống phân tán duy nhất.

Hệ thống phân tán này thường được tạo ra bằng cách chia nhỏ một ứng dụng nguyên khối lớn và mục đích là một tập hợp microservice tiếp tục hoạt động cùng nhau như một ứng dụng duy nhất. Cách thiết kế này thường không có hướng tới phục vụ nhiều hệ thống cùng một lúc.

Không giống như SOA, microservice thường tồn tại ngầm. Chúng không được phát hiện tại thời gian chạy và không yêu cầu trung gian. Bên tiêu thụ biết những dịch vụ này và do đó không yêu cầu bản mô tả. Tất nhiên, không phải là không cần khám phá, trong nhiều một số trường microservice cũng cần được khám phá.

Một số hệ thống microservice phức tạp có thể cần áp dụng một số mức độ khám phá dịch vụ cụ thể để làm cho các dịch vụ này trở nên linh hoạt hơn và mạnh mẽ hơn. Vấn đề là mô hình kiến trúc như vậy không cần thiết. Một hệ thống microservice mà tất cả các microservice đều nhận biết được nhau bằng cách sử dụng các tệp cấu hình tương đối đơn giản là hoàn toàn khả thi và có thể là một giải pháp hiệu quả. Các nhóm có thể khám phá microservice nào có sẵn để sử dụng tại thời điểm phát triển bằng cách sử dụng danh mục, bản đăng ký hoặc công cụ quản lý API.

Tuy nhiên, kiến trúc SOA và microservice chia sẻ nhiều nguyên tắc, mẫu và mô hình lập trình chung. Xét cho cùng, kiến trúc microservice là một loại SOA, ít nhất là theo một cách thô sơ.

Có thể xem microservice như là phần mở rộng/chuyên môn hóa của SOA, trong đó ranh giới khu vực chức năng được sử dụng để xác định các mô hình miền và gán chúng cho các nhóm có thể chọn ngôn ngữ phát triển, khuôn khổ và chi tiết triển khai của riêng. Tương tự, lắp ráp, điều phối, giám sát và quản lý dịch vụ là mối quan tâm thực sự và là yêu cầu trong các hệ thống microservice bên cạnh các hệ thống SOA truyền thống hơn.

Là hệ quả trực tiếp của quá nhiều phạm vi, tiêu chuẩn và công cụ mà SOA đã có trong quá khứ, kiến trúc microservice tập trung vào việc giải quyết một vấn đề duy nhất. Nó chủ yếu tập trung vào việc từng bước phát triển một ứng dụng lớn, nguyên khối thành một hệ thống vi dịch vụ phân tán dễ quản lý, phát triển và triển khai hơn bằng cách sử dụng các phương pháp tích hợp liên tục. Microservice đang tham gia vào cấu thành một hệ thống không nên lại được sử dụng bởi một hệ thống khác.

Mặc dù nhấn mạnh nhiều vào việc tái sử dụng, việc tái sử dụng này không xảy ra ở cấp độ dịch vụ. Tất cả microservice trong một hệ thống dịch vụ vi mô chủ yếu được điều khiển bởi một ứng dụng chính. Nếu muốn, những microservice này có thể được sử dụng lại bởi các ứng dụng hiện có hoặc mới khác để đạt được những lợi ích tương tự.

Không khó để hiểu sự phân hóa này diễn ra như thế nào. SOA được thiết kế với mục tiêu giải quyết các vấn đề rất phức tạp về kiến trúc doanh nghiệp, với mục tiêu tạo điều kiện cho khả năng tái sử dụng ở mức độ cao.

Ngược lại, kiến trúc microservice được chấp nhận bởi các công ty đang cố gắng mở rộng một thuộc tính web đơn lẻ thành quy mô web rộng hơn, cho phép phát triển liên tục, làm cho nhóm kỹ thuật hoạt động hiệu quả hơn và tránh bị khóa công nghệ.

Là hệ quả trực tiếp của phạm vi tập trung và hạn chế hơn, đồng thời nhấn mạnh vào sự phát triển gia tăng từ hệ thống truyền thống sang hệ thống microservice, microservice ngày càng hấp dẫn các doanh nghiệp có đầu tư cơ sở hạ tầng đáng kể. Phương pháp này hứa hẹn ít tốn

kém hơn và mang tính thử nghiệm hơn, trả tiền khi bạn chuyển đổi. Ngược lại, SOA truyền thống thường yêu cầu cam kết tài chính và kiến trúc trước nghiêm túc hơn nhiều.

Có thể nói, cả kiến trúc SOA và microservice đều là kiến trúc dịch vụ, bởi vì cả hai đều xử lý các hệ thống phân tán của các dịch vụ giao tiếp qua mạng. Tuy nhiên, kết quả thực tế khá khác biệt, bởi vì trọng tâm của SOA là khả năng tái sử dụng và khám phá, trong đó trọng tâm của microservices là thay thế một ứng dụng nguyên khối duy nhất bằng một hệ thống dễ dàng quản lý và phát triển dần dần.

5.4 BÀI TẬP

1. Kiến trúc hướng dịch vụ (SOA) là gì? Nêu các đặc trưng của kiến trúc hướng dịch vụ?
2. Kiến trúc RESTful là gì? Nêu các thao tác chính trong kiến trúc RESTful?
3. Kiến trúc vi dịch vụ microservice là gì ? So sánh, phân tích điểm giống và khác giữa SOA và Microservice.

CHƯƠNG 6
ONTOLOGY VÀ OWL

6.1 KHÁI NIỆM BẢN THỂ (ONTOLOGY) VÀ TRI THỨC

Một bản thể (ontology) là một loại biểu diễn tri thức mô tả các khái niệm của một vài lĩnh vực. Một bản thể đặc tả một danh mục các từ vựng bao gồm các thuật ngữ chính và các mối liên hệ ngữ nghĩa, và các luật suy diễn.

Trong thuật ngữ chung, một biểu diễn là một hệ thống các cụm từ, một bản thể cần áp dụng cho các thông tin được lưu trữ và thao tác, cho các nỗ lực tri thức. Chẳng hạn, chúng ta có một bản thể cho các khái niệm bảo mật, nó sẽ được hình thức hóa các thuật ngữ như vai trò, ủy nhiệm, xác thực, phân quyền v.v.. Một bản thể như vậy có thể được sử dụng để so sánh các sách tiếp cận bảo mật và có thể giúp chúng kết hợp với nhau.

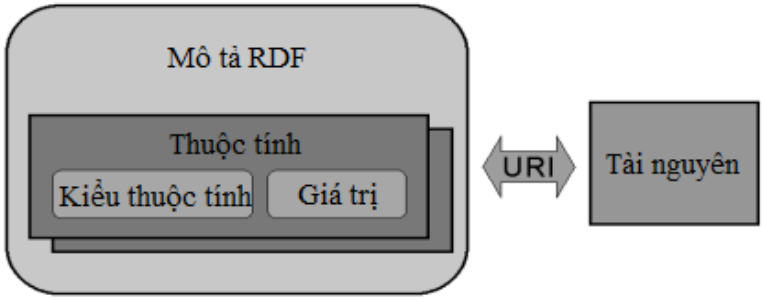
Một bản thể là một mô hình tính toán trong một vài lĩnh vực của thể giới thực. Nó thường được biết đến như một mạng ngữ nghĩa (semantic network), một đồ thị mà các nút là các khái niệm hoặc các đối tượng riêng biệt và các cạnh là mối quan hệ hoặc kết hợp giữa các khái niệm. Mạng ngữ nghĩa này được tăng cường bởi các thuộc tính, ràng buộc, chức năng, qui định (rules), chúng chi phối các ứng xử của các khái niệm.

Một cách hình thức bản thể là một hợp đồng về mức khái niệm được chia sẻ, nó bao gồm các nền tảng cho mô hình hóa tri thức và hợp lý về biểu diễn một vài lý thuyết về một lĩnh vực cụ thể. Các định nghĩa gắn kết với các tên của thực thể trong một đoạn (các lớp, các mối quan hệ, các hàm, và các đối tượng khác) cùng với văn bản mà con người có thể đọc được. mô tả các nghĩa của tên, và các tiên đề hình thức mà rang buộc thông dụng và hình thức sử dụng các tên này.

Đối với từng hệ thống thông tin trên internet, các bản thể có thể được sử dụng để tổ chức các từ khóa và các khái niệm cơ sở dữ liệu bằng việc thiết lập mối quan hệ ngữ nghĩa giữa các từ khóa hoặc giữa các bảng và trường dữ liệu trong cơ sở dữ liệu. Các quan hệ ngữ nghĩa cung cấp cho người dùng một cái nhìn tổng thể về không gian thông tin trong lĩnh vực quan tâm.

6.2 NGÔN NGỮ MÔ TẢ NGUỒN RDF

RDF (Resource Description Framework) là nền tảng chuẩn cho việc trao đổi dữ liệu trên Web. RDF có các đặc tính làm cho việc phối hợp (merging) dữ liệu trở nên dễ dàng hơn thậm chí dữ liệu được mô tả dưới các lược đồ khác nhau. RDF cũng hỗ trợ việc tiến triển các lược đồ theo thời gian mà không cần phải thay đổi dữ liệu. RDF mở rộng các cấu trúc liên kết của Web để sử dụng URI cho các tên của các quan hệ giữa các đối tượng và 2 đầu liên kết. Với mô hình đơn giản này, nó cho phép các dữ liệu có cấu trúc và bán cấu trúc có thể được trộn lẫn với nhau để chia sẻ cho các ứng dụng khác nhau. Các dạng cấu trúc liên kết bao gồm đồ thị có hướng và được gán nhãn với các cạnh biểu diễn mỗi quan hệ giữa hai tài nguyên và tài nguyên được mô tả bởi các nút của đồ thị.



Hình 5.1 Mô hình RDF

Các ngôn ngữ truy vấn RDF có thể được phân thành ba nhóm dựa trên mô hình dữ liệu, đặc trưng, lược đồ biểu diễn dữ liệu hoặc cách thức truy vấn. Các ngôn ngữ này coi RDF như các bộ ba dữ liệu mà lờ đi các lược đồ hay thậm chí lờ đi các thông tin về ontology nếu không được trình bày trong RDF. Một trong các ngôn ngữ RDF là RQL có hỗ trợ truy vấn dữ liệu và lược đồ.

Một tài liệu RDF là một tập hợp các câu lệnh (statements), mỗi câu lệnh được là một bộ ba chủ ngữ (subject), và một bổ ngữ (object). Chủ ngữ và vị ngữ là một tài nguyên (resource), và bổ ngữ có thể là một tài nguyên hoặc một từ (literal). Vị ngữ được gọi là một tài sản (property). Thuật ngữ RDF cho một vị ngữ là rdf:Property. Một câu lệnh được đặc trưng bởi chủ ngữ của nó và nó liên quan đến bổ ngữ thông qua vị ngữ của nó.

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://www.wiley.com/SOC">
     <dc:title>Service-Oriented Computing</dc:title>
     <dc:creator>Munindar</dc:creator>
     <dc:creator>Michael</dc:creator>
     <dc:publisher>Wiley</dc:publisher>
   </rdf:Description>
</rdf:RDF>
```

Hình 5.2: Ví dụ về một tài liệu RDF

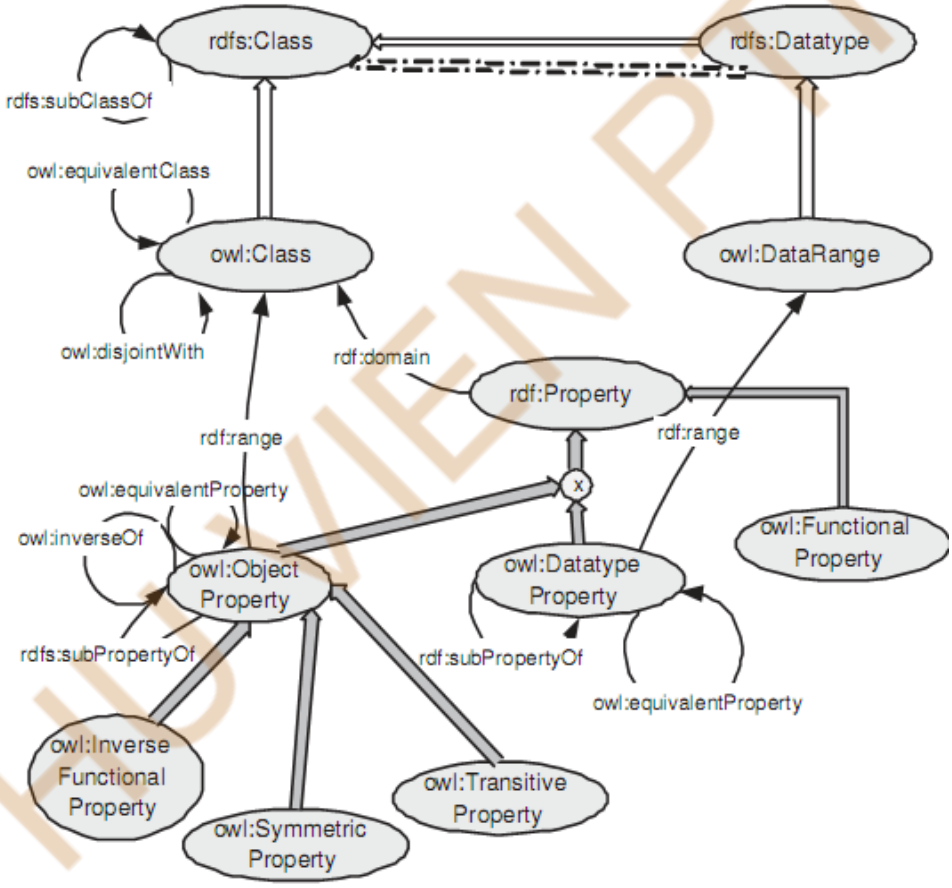
6.3 BẢN THỂ HỌC OWL

RDF (Resource Description Framework) là nền tảng chuẩn cho việc trao đổi dữ liệu trên Web. RDF có các đặc tính làm cho việc phối hợp (merging) dữ liệu trở nên dễ dàng hơn thậm chí dữ liệu được mô tả dưới các lược đồ khác nhau. RDF cũng hỗ trợ việc tiến triển các lược đồ theo thời gian mà không cần phải thay đổi dữ liệu. RDF mở rộng các cấu trúc liên kết của Web để sử dụng URI cho các tên của các quan hệ giữa các đối tượng và 2 đầu liên kết. Với mô hình đơn giản này, nó cho phép các dữ liệu có cấu trúc và bán cấu trúc có thể được trộn lẫn với nhau để chia sẻ cho các ứng dụng khác nhau. Các dạng cấu trúc liên kết bao gồm đồ thị có hướng và được gán nhãn với các cạnh biểu diễn mỗi quan hệ giữa hai tài nguyên và tài nguyên được mô tả bởi các nút của đồ thị.

6.3.1 Xác định lớp dựa trên OWL

OWL (The Web Ontology Language) là một họ ngôn ngữ dùng để mô tả tri thức hoặc các ngôn ngữ bản thể cho các bản thể tác giả hoặc cơ sở tri thức. Ngôn ngữ này được đặc trưng bởi ngữ nghĩa hình thức. OWL là một ngôn ngữ đánh dấu dùng để xuất bản và chia sẻ dữ liệu trên Internet thông qua những bản thể OWL là phần mở rộng về từ vựng của RDF một dự án được hỗ trợ bởi W3C. OWL được xem như là một kỹ thuật trọng yếu để cài đặt cho Semantic Web trong tương lai.

OWL cung cấp các khả năng mô tả lớp và thuộc tính theo dạng lô gic mô tả với các phần tử trong các biểu thức sử dụng các phép toán Boolean AND, OR, NOT và các thuộc tính ràng buộc. Dạng đặc biệt của ràng buộc bao gồm sự tồn tại của các thuộc tính. Chẳng bạn, người cha có thể được xác định là người trưởng thành và có ít nhất một con.



Hình 5.3: Mô hình các đối tượng và các quan hệ của OWL

Các lớp tương ứng với các tập các đối tượng. Các loại đối tượng được phân biệt từ các loại dữ liệu. Một biến thể hiện của một lớp đối tượng không thể là biến thể hiện của kiểu dữ liệu. OWL có một lớp có tên là owl:Class được định nghĩa là lớp con của rdfs:Class. Tất cả các lớp đối tượng OWL là thành viên của owl:Class. OWL cung cấp nhiều loại dữ liệu, bao gồm cả những loại dữ liệu được đặc tả dựa trên RDF, các phần tử của lượt đồ RDF và kiểu liệt kê dựa trên đối tượng. Hình 5.4 là một ví dụ:

```
<owl:Class rdf:ID="Reptile">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf rdf:resource="#OxygenUser"/>
</owl:Class>
```

Hình 5.4: Ví dụ về một tài liệu OWL đơn giản

Lớp owl:Thing là đỉnh (top) của lớp phân cấp. OWL cũng gồm lớp owl:NoThing mà không có các biến thể hiện. Do đó nó là lớp con của mọi lớp. Một cách hình thức, owl:Thing và owl:NoThing bổ trợ cho nhau. Chúng ta có thể khai báo các thành viên của mô tả lớp bởi việc sử dụng tên lớp như tên các phần tử; đây là trường hợp đặc biệt của qui ước cú pháp RDF/XML mà rdf:type được sử dụng như tên phần tử:

```
<Reptile rdf:ID="BobsLizard"/>
<owl:Thing rdf:ID="BobsLizardCage"/>
```

6.3.2 Xác định các tính chất dựa trên OWL

Các tính chất liên quan đến các cặp cá nhân. Thuộc tính đối tượng, biến thể hiện của owl:ObjectProperty các biến thể hiện liên quan đến hai lớp. Thuộc tính kiểu dữ liệu, biến thể hiện của owl:DatatypeProperty, liên quan đến một biến thể hiện của một lớp với một biến thể hiện của một kiểu dữ liệu. Ngược lại, trong RDF ta có thể khai báo owl:ObjectProperty hoặc owl:DatatypeProperty.

```
<owl:ObjectProperty rdf:ID="livesIn">
  <rdfs:domain rdf:resource='#Animal'/>
  <rdfs:range rdf:resource='#Locale'/>
  <rdfs:subPropertyOf rdf:resource='#hasHabitat'/>
  <owl:inverseOf rdf:resource='#hasDenizen'/>
  <owl:equivalentProperty rdf:resource='#hasHome'/>
</owl:ObjectProperty>
```

Hình 5.5: Ví dụ về một tài liệu owl với khai báo owl:ObjectProperty

Vùng (domain) và khoảng (range) là các ràng buộc toàn cục do chúng áp dụng đến một tính chất độc lập của lớp với những biến thể hiện mà tích chất đó được áp dụng. Vùng của một tính chất phải là biến thể hiện của owl:Class và owl:Thing trừ khi được mô tả chi tiết. Khoảng của một tính chất đối tượng là biến thể hiện của owl:Class và owl:Thing trừ khi được mô tả chi tiết. Tuy nhiên, khoảng của một tính chất kiểu dữ liệu là owl:DataRange, đó là lớp con của rdfs:DataType.

OWL cho phép nhiều điều kiện (assertions) về vùng và khoảng của thuộc tính. Những điều kiện này được thông dịch giao nhau (interpreter conjunctively). Nói cách khác, những thứ trong vùng là một thuộc tính nếu chúng là phép giao của tất cả các vùng; khoảng cũng có tính chất tương tự như vùng.

Phần tử rdf:Property tham chiếu tới một tên thuộc tính như là một URI. Như đã giải thích ở trên, biến thể hiện của một thuộc tính là các cặp các cá thể (individuals): cá thể thứ nhất là một đối tượng và cá thể thứ hai là một giá trị dữ liệu. Một phần tử rdf:Property có thể bao gồm:

- Không hoặc hơn rdfs:subPropertyOf phần tử, mỗi phần tử có một tên. Mỗi cặp cá thể là biến thể hiện của thuộc tính được đặt tên trong phần tử chính rdf:Property phải là một biến thể hiện của thuộc tính được đặt tên trong mỗi rdfs:subPropertyOf phần tử. Một owl:ObjectProperty không thể là thuộc tính con của owl:DatatypeProperty và ngược lại.
- Không hoặc hơn rdfs:domain phần tử. Thành phần đầu tiên của mỗi biến thể hiện mà thuộc tính áp dụng phải trong vùng trạng thái. Nhiều phần tử vùng do đó thông dịch giao nhau.
- Không hoặc hơn rdfs:range phần tử. Thành phần thứ 2 của mỗi biến thể hiện mà thuộc tính áp dụng phải trong vùng trạng thái. Nhiều phần tử vùng do đó thông dịch giao nhau.
- Không hoặc hơn owl:equivalentProperty, xác nhận tính tương đương của 2 thuộc tính. Đây là thuộc tính con của rdfs:subPropertyOf. Do đó, nó phải thỏa mãn ràng buộc đã được đề cập ở trên về việc không liên quan giữa owl:ObjectProperty với owl:DatatypeProperty.
- Không hoặc hơn owl:inverseOf phần tử với tên thuộc tính được đảo ngược. Tất cả các thuộc tính phái có owl:Class như vùng của chúng. Chú ý là owl:inverseOf chỉ áp dụng cho owl:ObjectProperty.

6.3.3 Ba mô hình với OWL

OWL được xem như là một kỹ thuật trọng yếu để cài đặt cho Semantic Web trong tương lai. OWL được thiết kế đặc biệt để cung cấp một cách thức thông dụng trong việc xử lý nội dung thông tin của Web. Ngôn ngữ này được sẽ cho phép các hệ thống máy tính có thể hiểu được ngữ nghĩa thông tin do tài liệu OWL được viết bởi XML. Hiện nay có ba mô hình ngôn ngữ OWL: OWL Lite, OWL DL (description logic), và OWL Full. Các lớp ngôn ngữ này tăng độ phức từ đơn giản đến phức tạp.

- *OWL Lite*: hỗ trợ cho biểu diễn dạng thông tin cần sự phân lớp theo thứ bậc và các ràng buộc đơn giản. Điều này cho phép cung cấp các công cụ hỗ trợ OWL Lite dễ dàng hơn so với các phiên bản OWL khác bản khác.
- *OWL DL (OWL Description Logic*): hỗ trợ cho việc duy trì tính tính toán toàn vẹn và tính quyết định OWL DL bao gồm tất cả các cấu trúc của ngôn ngữ OWL, ví dụ: Trong khi một lớp có thể là một lớp con của rất nhiều lớp, một lớp không thể là một thể hiện của một lớp khác. OWL DL cũng phù hợp tương ứng với logic mô tả.
- *OWL Full* không cần đảm bảo sự tính toán của các biểu thức. Ví dụ, trong OWL Full, một lớp có thể được xem xét đồng thời như là một tập của các cá thể và như là một cá thể trong chính bản thân nó. OWL Full cho phép một bản thể tăng cường ý nghiã của các từ vựng được định nghĩa trước (RDF hoặc OWL). OWL Full là lớp ngôn ngữ phức tạp OWL phức tạp nhất trong 3 lớp ngôn ngữ.

Ngôn ngữ OWL Lite giới hạn các giá trị trong tập là 0 hoặc 1. Đối với tập nhỏ nhất, đặc tả rỗng (nothing) tương đương với đặc tả 0, vì vậy chỉ cần thiết để đặt tả tham gia bắt buộc. Đối với tập lớn nhất, đặc tả rỗng tương đương với không giới hạn (unbounded). Nếu tập lớn nhất là 1 nghĩa là đó là một giá trị kết hợp lớn nhất. Nhìn chung, giới hạn tập lớn nhất là 0 thì không nên sử dụng vì nó cấm việc tham dự.

6.3.4 Ví dụ

Ta xem xét ví dụ đơn giản về sinh viên đăng kí học và các môn học được quản lý bởi các khoa. Khoa khoa học máy tính là một khoa. Mỗi sinh viên phải học ít nhất một môn, và một sinh viên chính qui (full time) phải học từ 3 đến 5 môn. Để biểu diễn tình huống này trong OWL.

```
<owl:Class rdf:ID="Student"/>
<owl:Class rdf:ID="Course"/>
<owl:Class rdf:ID="Department"/>

<Department rdf:ID='CS'/>

<owl:ObjectProperty rdf:ID='takes'>
  <rdfs:domain rdf:resource='#Student'/>
  <rdfs:range rdf:resource='#Course'/>
</owl:ObjectProperty>

<owl:InverseFunctionalProperty rdf:ID='offers'>
  <rdfs:domain rdf:resource='#Department'/>
  <rdfs:range rdf:resource='#Course'/>
</owl:InverseFunctionalProperty>

<owl:ObjectProperty rdf:ID='offeredBy'>
  <inverseOf rdf:resource='#offers'/>
<!— implies that offeredBy is a FunctionalProperty —>
</owl:ObjectProperty>
```

Hình 5.6 Tài liệu OWL mô tả quản lý học tập của sinh viên

Hình 5.6 ở trên đã mô tả tất cả các ràng buộc ngoại trừ một sinh viên phải học ít nhất một môn. Tài liệu OWL dưới đây bổ sung ràng buộc đó.

```
<owl:Restriction>
  <owl:onProperty rdf:resource='#takes'/>
  <owl:minCardinality
     rdf:datatype='&xsd;#nonNegativeInteger'>
    1
  </owl:minCardinality>
</owl:Restriction>
```

Tuy nhiên, khai báo một lớp nặc danh mới của các sinh viên học ít nhất một môn. Khi biết domain của takes là Student, ta có thể tham chiếu lớp nặc danh ở trên là lớp con của Student. Tuy nhiên, điều đó cũng chưa nói nên một sinh viên phải học ít nhất một môn. Để thực hiện yêu cầu này, ta phải bổ sung sinh viên là một lớp con của lớp nặc danh ở trên. Một trong những tiên đề của OWL được sử dụng:

```
<owl:Class rdf:about="#Student">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='#takes'/>
      <owl:minCardinality
        rdf:datatype='&xsd;#nonNegativeInteger'>
       1
      </owl:minCardinality>
    </owl:Restriction>
  <rdfs:subClassOf>
</owl:Class>
```

Hình 5.7: Sinh viên là một lớp con của lớp nặc danh

Như đã bàn ở trên, ta có thể trình bầy định nghĩa các sinh viên full-time theo biểu thức phức tạp sau đây:

```
<owl:Class rdf:ID=”FullTimeStudent”
  <owl:intersectionOf rdf:parseType='Collection'>
    <rdfs:Class rdf:about='#Student'/>
    <owl:Restriction>
      <owl:onProperty rdf:resource='#takes'/>
      <owl:minCardinality
        rdf:datatype='&xsd;#nonNegativeInteger'>
       3
      </owl:minCardinality>
      <owl:maxCardinality
        rdf:datatype='&xsd;#nonNegativeInteger'>
       5
      </owl:maxCardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Hình 5.8: Lớp Fulltime Student

Ta thêm lớp về các môn học được cung cấp bởi khoa Khoa học máy tính (CS)

```
<owl:Class rdf:ID="CSCourse">
  <owl:intersectionOf rdf:parseType='Collection'>
    <rdfs:Class rdf:about='#Course'/>
    <owl:Restriction>
      <owl:onProperty rdf:resource='#offeredBy'/>
      <owl:hasValue rdf:resource='#CS'/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Hình 5.9: Lớp Fulltime Student

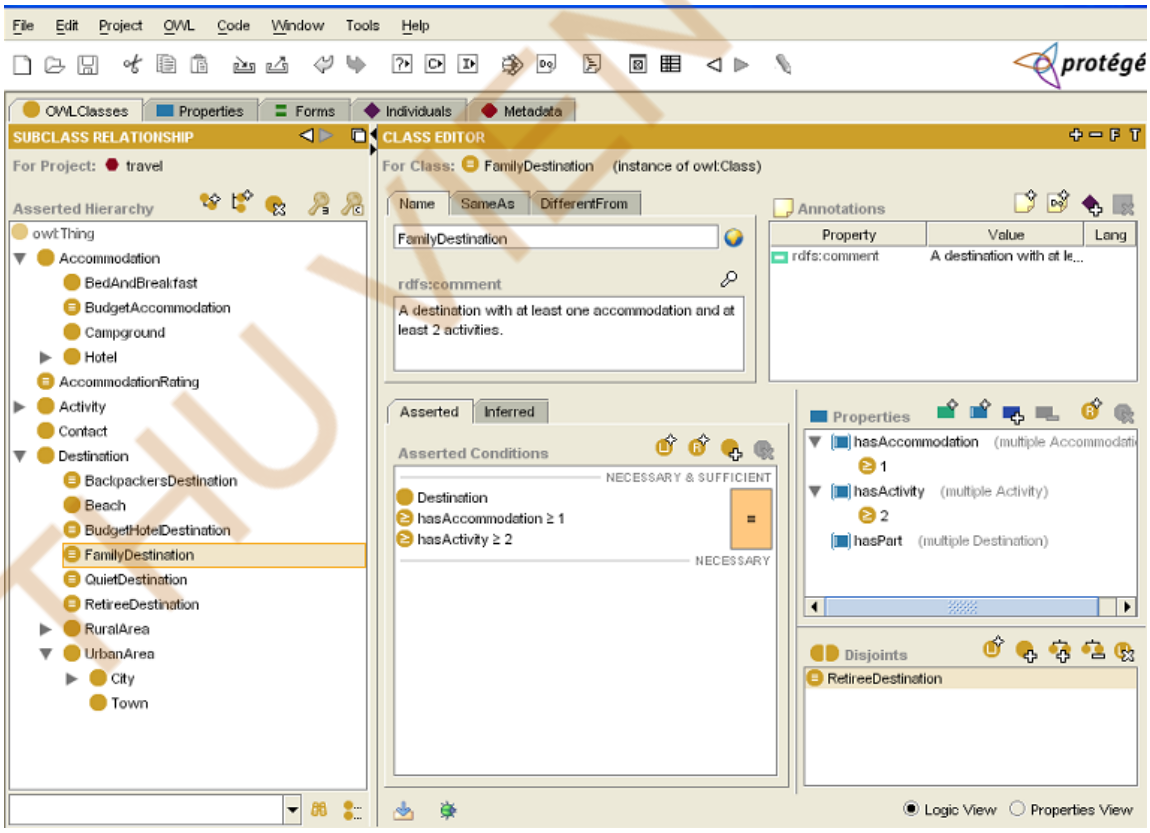
Các sinh viên full-time ngành khoa học máy tính có thể được định nghĩa là các sinh viên chỉ học các môn của khoa Khoa học Máy tính.

```
<owl:Class rdf:ID="CSFullTimeStudent">
  <owl:intersectionOf rdf:parseType='Collection'>
    <rdfs:Class rdf:about='#FullTimeStudent'/>
    <owl:Restriction>
      <owl:onProperty rdf:resource='#takes'/>
      <owl:allValuesFrom rdf:resource='#CSCourse'/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Hình 5.10: Lớp CSFulltimeStudent

6.4 CÔNG CỤ PROTÉGÉ CHO XÂY DỰNG OWL

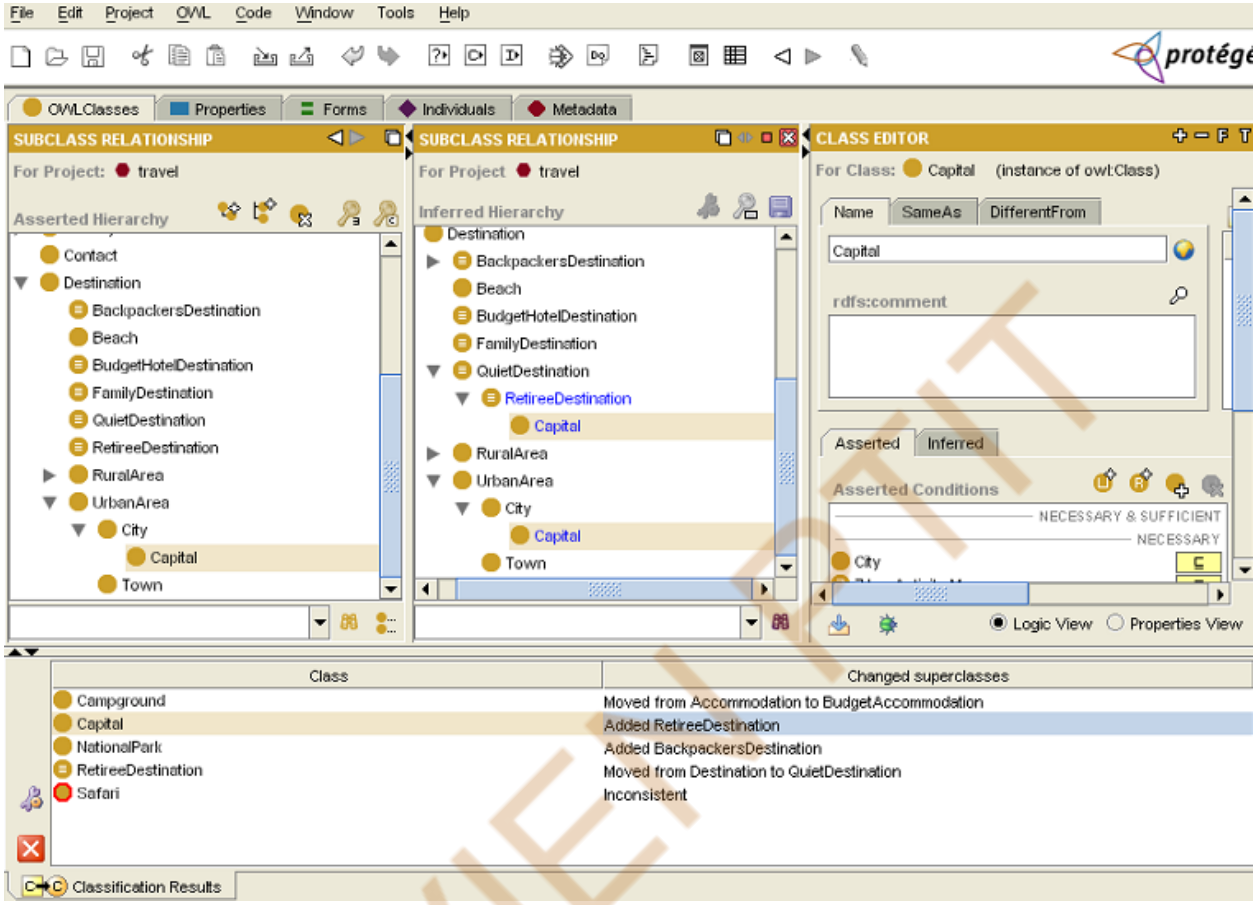
Protégé là công cụ soạn thảo (editor) bản thể mã nguồn mở và là một hệ thống thu nhận tri thức được phát triển bởi trường đại học Stanford. Protégé cung cấp một giao diện đồ họa cho người dùng để định nghĩa các bản thể. Protégé tích hợp các bộ phân loại suy dẫn (deductive classifiers) để kiểm nghiệm tính nhất quán của các mô hình và suy dẫn thông tin mới dựa trên việc phân tích trên bản thể. Hình 5.11 dưới đây là minh họa về giao diện của công cụ Protégé.



Hình 5.11: Giao diện công cụ Protégé

Protégé OWL là một plug-in của nền tảng phát triển Protégé cho bản thể (Protégé ontology development platform). Protégé OWL cho phép người sử dụng soạn thảo các bản thể bằng ngôn ngữ OWL và để sử dụng các bộ lô gic mô tả để duy trì tính nhất quán trong các bản thể của chúng. Protégé OWL được phát triển bằng Java, và nó chạy trên nhiều loại nền tảng

khác nhau, và nó có cộng đồng đông đảo người dùng và trở thành bộ soạn thảo chuẩn cho OWL. Hình 5.12 minh họa tiện ích phân loại của Protégé OWL của ứng dụng Travel [3].



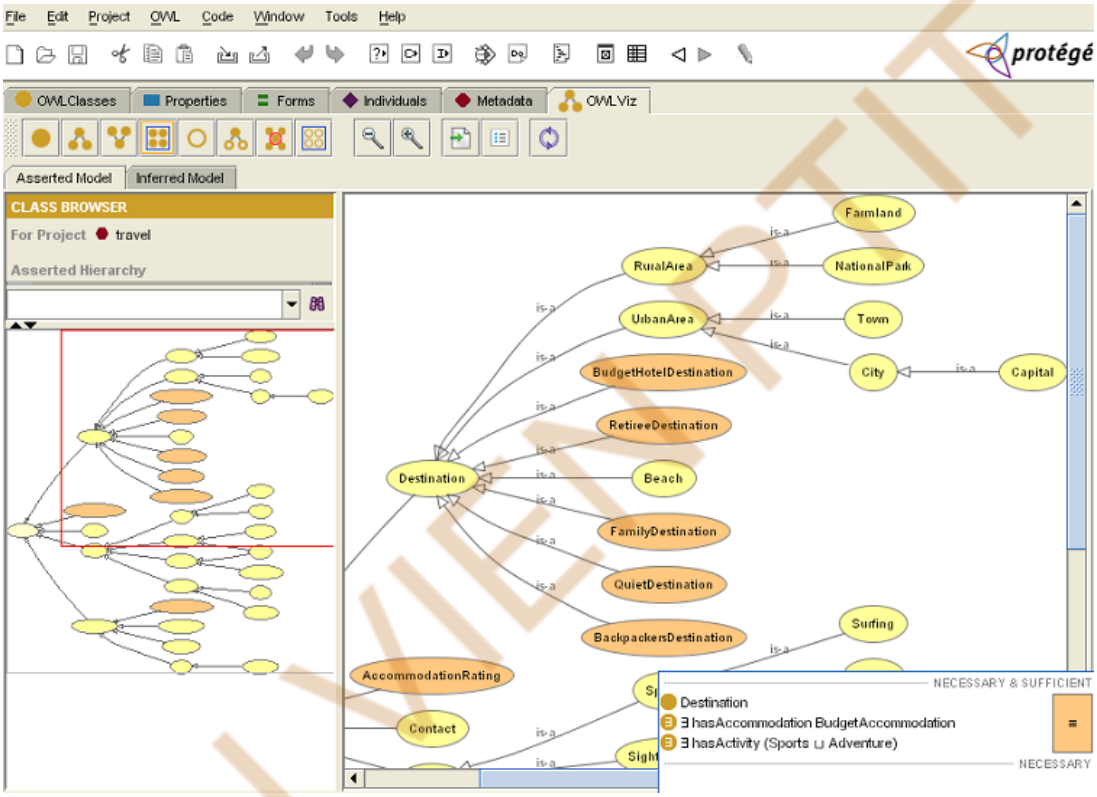
Hình 5.12: Tiện ích phân loại của công cụ Protégé [3]

Protégé OWL cung cấp các tính năng sau đây làm cho nó trở thành rất có ích cho việc xây dựng các bản thể bằng OWL và các ứng dụng thông minh mà sử dụng các bản thể này:

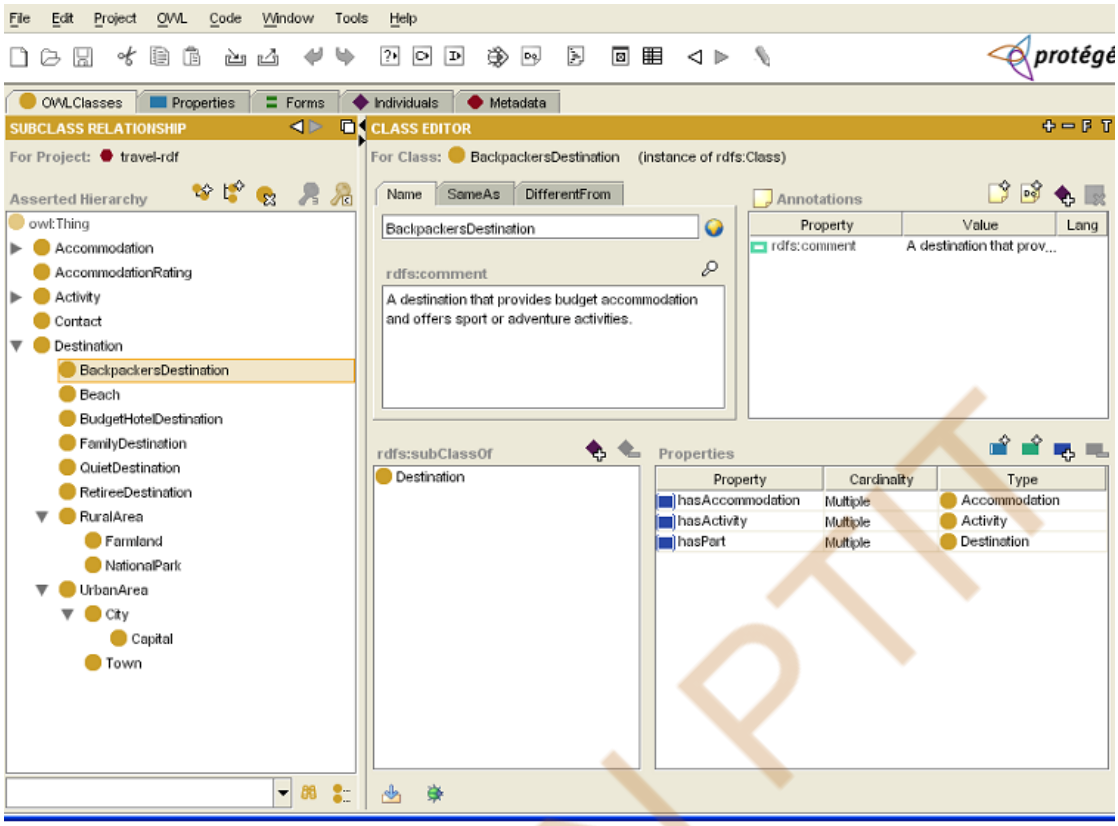
- Giao diện đồ họa cho người dùng (GUI) và API: Protégé OWL được xây dựng dựa trên mô hình nền tảng tri thức Protégé và sử dụng Protégé GUI để soạn thảo các lớp, thuộc tính và các biến thể hiện. Nó cung cấp các thư viện API cho phép người dùng tích hợp Protégé OWL vào ứng dụng của họ. Hình 5.13 minh họa giao diện plug-ins của Protégé OWL của ứng dụng Travel [3].
- Bộ soạn thảo đồ họa cho các biểu thức OWL lô gic: Protégé OWL cung cấp bộ soạn thảo biểu thức dễ sử dụng cho phép người dùng thao tác trên các biểu thức một cách dễ dàng bằng chuột và bàn phím. Nó cũng sử dụng giao diện đồ họa hướng đối tượng để định nghĩa các lớp. Bộ soạn thảo này cũng hỗ trợ các thao tác kéo thả và copy/paste. Hình 5.14 minh họa giao diện của bô soạn thảo Protégé OWL.
- Hỗ trợ thực hiện các thao tác phức tạp: với tính năng hỗ trợ wizard cho các mẫu bản thể công nghệ như tạo các nhóm lớp, tạo tập hợp các lớp không giao nhau, tạo ra ma trận thuộc tính cho nhiều giá trị thuộc tính, và tạo các giá trị phân hoạch.
- Trực tiếp truy cập đến các bộ suy diễn: Protégé OWL cung cấp khả năng tuy cập đến các bộ phân loại hiệu năng cao như Racer. Giao diện người dùng hỗ trợ 3 loại suy diễn: (1) kiểm tra tính nhất quán; (2): phân loại; và (3): phân loại các biến thể hiện. Do

Protégé OWL xây dựng các hệ thống Protégé, nên các đặc trưng hữu ích sau cũng có sẵn:

- Tự động sinh ra form: Protégé OWL có thể tự động sinh ra một giao diện người dùng để yêu cầu dữ liệu từ các lớp định nghĩa. Đặc tính này rất có ích để thu thập tri thức.
- Hỗ trợ đa người dùng: Protégé OWL cung cấp khả năng đa người dùng cho điểm vào của đồng bộ tri thực.
- Hỗ trợ nhiều định dạng lưu trữ: Protégé OWL có thể mở rộng phần back end cho phép lưu trữ nhiều các dạng file có định dạng khác nhau như: Clips, XML, RDF, và OWL.



Hình 5.13: Giao diện plug-ins của Protégé OWL với ứng dụng Travel [3]



Hình 5.14: Bộ soạn thảo của Protégé OWL [3]

Hiện nay Protégé OWL có phiên bản:

- Phiên bản chạy trên Web: cho phép người sử dụng có thể thực hiện việc biên tập, xây dựng bản thể trực tuyến tại địa chỉ: http://protege.stanford.edu/products.php#web-protege . Ngoài ra. Người dùng có thể tải bộ cài về từ địa chỉ: https://github.com/protegeproject/webprotege/releases
- Phiên bản chạy trên Desktop: cho phép người sử dụng có thể thực hiện việc biên tập, xây dựng bản trên máy tính cá nhân. Bộ cài có thể được tải từ: http://protege.stanford.edu/products.php#desktop-protege

6.5 BÀI TẬP

1. Phát biểu nào sau đây về các tài liệu Web ngữ nghĩa là KHÔNG đúng:

   A. Chúng có thể tham chiếu đến các thuật ngữ (term) mà không cần đến bản thể (ontology);

   B. Chúng được viết bằng XML;

   C. Chúng được viết bằng RDF;

   D. Chúng có thể tham chiếu đến nhiều bản thể;

   E. Chúng có thể định nghĩa các bản thể mới.

   Hãy giải thích về sự lựa chọn của bạn

2. Lựa chọn nào dưới đây mô tả tốt nhất về sự khác nhau giữa thuộc tính rdf:ID và rdf:about

   A. Chúng tương đương nhau;

   B. rdf:ID được sử dụng khi định nghĩa một đối tượng, còn rdf:about được sử dụng khi tham chiếu đến một đối tượng;

   C. rdf:ID được sử dụng cho mô tả tài nguyên còn rdf:about được sử dụng cho vị ngữ (predicates);

   D. rdf:ID có thể có giá trị còn giá trí của rdf:about phải là URN

   E. rdf:ID không hỗ trợ sử dụng namespaces nhưng rdf:about có hỗ trợ sử dụng namespaces

3. Tài liệu RDF nào dưới đây tương ứng với câu trong tiếng Anh "Anything that is a student is a person?"

   A.

```
<!-- rdfs is the URI for the RDF Schema namespace -->
<rdf:Description rdf:ID="student">
  <rdf:type
     resource="&rdfs;#Class"/>
  <rdfs:subClassOf rdf:resource="#person"/>
</rdf:Description>
```

   B.

```
<rdf:Description>
  <student>
    <rdfs:subClassOf rdf:resource="#person"/>
  </student>
</rdf:Description>
```

   C.

```
<rdf:Description rdf:about="student">
  <rdfs:isA rdf:resource="#person"/>
</rdf:Description>
```

   D.

```
<rdf:Class rdf:ID="student">
  <rdfs:subClassOf rdf:resource="#person"/>
</rdf:Description>
```

   F. Không có mô tả nào ở trên và bạn tự viết mô tả RDF cho câu đó.

4. OWL có thể sử dụng cho:

A. Tạo ra các bản thể

B. Mô tả các dịch vụ Web

C. Mô tả các ứng dụng

D. Định nghĩa các kiểu lược đồ XML

E. Tạo ra các thông điệp multicast SOAP

5. Giả sử định nghĩa một thuộc tính parentOf liên quan đến các động vật và một lớp Parent (bố, mẹ) và giới hạn parentOf có số lượng nhỏ nhất là 1. Hãy viết một mô tả OWL cho grandparent (ông, bà).

6. Hãy tạo một tài liệu OWL để mô tả tình huống sau đây cho một trường đại học. Giả sử các lớp student, course, và department với các thuộc tính takes và offeredBy. Bạn có thể định nghĩa các lớp và thuộc tính phụ nếu bạn muốn. Sử dụng các cú pháp RDF-XML:

- Định nghĩa một danh sách trích ngang các môn học mà được cung cấp (offered) bởi hai hay nhiều hơn hai department;
- Một sinh viên thuộc về chính xác một department;
- Định nghĩa một sinh viên không may mắn mà học môn học đang được học bởi nhiều hơn 100 sinh viên khác.

CHƯƠNG 7
DỊCH VỤ WEB NGỮ NGHĨA VÀ OWL-S

Sự hội tụ của web ngữ nghĩa với điện toán hướng dịch vụ đà hình thành công nghệ dịch vụ web ngữ nghĩa (Semantic Web Services- SWS). Nhằm giải quyết những thử thách chính của dịch vụ web như tự động, trao đổi trong suốt, kết hợp ý nghĩa được thực thi bởi các tác tử phần mềm thông minh.

7.1 BIỂU DIỄN NGỮ NGHĨA CỦA DỊCH VỤ WEB

Mỗi nền tảng biểu diễn dịch vụ ngữ nghĩa có thể được đặc trưng bởi: (a) loại dịch vụ ngữ nghĩa được mô tả, (b) biểu diễn bằng ngôn ngữ gì, (c) cho phép loại suy diễn gì từ biểu diễn tóm tắt dịch vụ. Hơn nữa, ta cần phân biệt giữa một phần tóm tắt dịch vụ web (abstract Web Service), cái mà biểu diễn các thực thể tính toán của dịch vụ , và dịch vụ chi tiết (concrete service) như là một trong những biến thể hiện cung cấp những giá trị thực sự cho người dùng. Trong ngữ cảnh này, biểu diễn tóm tắt dịch vụ được xem như đầy đủ nhưng không nhất thiết phái luôn đúng: có thể những biến thể hiện dịch vụ chi tiết mà các mô hình biểu diễn của tóm tắt dịch vụ có thể không được phân phát bởi nhà cung cấp. Việc biểu diễn ngữ nghĩa cho dịch vụ Web có những thành phần sau:

4 *Dịch vụ ngữ nghĩa có chức năng hoặc không có chức năng* (functional and non-functional service semantics): Nhìn chung, chức năng của một dịch vụ có thể được mô tả là nó làm gì và nó làm như thế nào. Cả hai lĩnh vực của ngữ nghĩa có chức năng được quản lý bởi một hồ sơ dịch vụ (service profile) và mô hình tiến trình dịch vụ. Hồ sơ dịch vụ mô tả chữ ký của dịch vụ bao gồm các tham số đầu vào (I) và đầu ra (O), và nguyên nhân (preconditions-P) và kết quả (effects-E) nó được hỗ trợ để thực thi dịch vụ với một trạng thái cho trước, và một vài thông tin như tên dịch vụ, các lĩnh vực nghiệp vụ và nhà cung cấp. Mô hình tiến trình của dịch vụ đơn nguyên (atomic) và hợp nhất (composite) mô tả dịch vụ thực hiện như thế nào với dữ liệu và luồng điều khiển dựa trên một tập luồng công việc hoặc dạng điều khiển như tuần tự, phân chia + hợp (split+join), lựa chọn v.v..

   Điểm khác nhau giữa hồ sơ ngữ nghĩa và mô hình tiến trình ngữ nghĩa là các nền tảng biểu diễn cấu trúc dịch vụ web, trong khi sự khác nhau trong cách đặt tên và biểu diễn hình thức từng phần của dịch vụ ngữ nghĩa. Bao gồm sự khác nhau giữa biểu diễn dịch vụ không trạng thái (stateless IO) và có trạng thái (state-base) khi mô tả tập các biến thể hiện mad dịch vụ chi tiết cung cấp các giá trị cho người dùng. Ngữ nghĩa của dịch vụ không chức năng thường được mô tả như một mô hình chất lượng dịch vụ (QoS) bao gồm các rang buộc, mô hình chi phí (cost model) với các luật giá cả, từ chối, có sẵn, và các chính sách riêng biệt (privacy).

5 *Biểu diễn có cấu trúc của ngữ nghĩa của dịch vụ:* Một biểu diễn có cấu trúc ngữ nghía của dịch vụ được cung cấp bởi các bản thể và ngôn ngữ của dịch vụ như OWL-S (OWL service) và WSML (Web Service Markup Language) với lô gic hình thức. Cả OWL-S và WSML có thể cung cấp các biểu diễn hình thức với các chuẩn dựa trên luồng công việc của mô hình tiến trình dịch vụ. Thay vì đó, biểu diễn tóm tắt dịch vụ được sinh ra

(grounded) trong WSDL, mô hình tiến trình có thể được ánh xạ đến BPEL với ngữ nghĩa hình thức xác định

6 *Biểu diễn đơn (monolithic) ngữ nghĩa của dịch vụ:* Đặc tả hình thức ngữ nghĩa của dịch vụ bất khả thi đối với bất kỳ định dạng biểu diễn nào. Chẳng hạn, bởi tập xác định của các tiên đề khái niệm và vai trò trong một lô gic thích hợp. Do khả năng của dịch vụ được mô tả bởi một khái nhiệm dịch vụ đơn, biểu diễn ngữ nghĩa của dịch vụ này được gọi đơn và cho phép xác định quan hệ ngữ nghĩa giữa biểu diễn dịch vụ đầy đủ với lô gic hình thức dựa trên phép thỏa (satisfaction), hợp nhất (subsumption), và kế thừa (entailment). Tuy nhiên nó không cung cấp thêm thông tin về việc thực tế dịch vụ thực hiện như thế nào trong mô hình tiến trình hay biểu diễn của ngữ nghĩa không chức năng.

7 *Ngữ nghĩa của dữ liệu:* các tham số của hồ sơ dịch vụ của ngữ nghĩa phụ thuộc lĩnh vực (domain-dependent semantics) được mô tả trong các khái niệm và vai trò từ lĩnh vực, nhiệm vụ hoặc các bản thể ứng dụng. Những bản thể này được định nghĩa trong các ngôn ngữ ngữ nghĩa web như OWL, WSML hay SMRL. Nếu các bản thể khác nhau được sử dụng, các tác tử được giả định tự động giải quyết vấn đề không đồng nhất cho các phương tiện trao đổi trong suốt (interoperation) tốt hơn khám phá dịch vụ web và hợp nhất. Các quá trình so khớp trong bản thể thường được giới hạn trong các bản thể được mô tả trong cùng một ngôn ngữ.

8 *Suy diễn về biểu diễn ngữ nghĩa của dịnh vụ:* ý tưởng cơ bản của biểu diễn dịch vụ web là cho phép các tác tử hiểu rõ hơn về ngữ nghĩa chức năng và không chức năng thông qua các suy diễn lô gic. Với mục đích đó thì suy diễn lô gics được đi kèm với nền tảng biểu diễn ngữ nghĩa của dịch vụ. Hơn nữa các biểu thức khái niệm được sử dụng để mô tả ngữ nghĩa dữ liệu với các tham số đầu vào và đầu ra được giả định để xây dựng từ các khái niệm cơ bản và các vai trò từ ứng dụng hình thức hoặc các bản thể lĩnh vực mà người yêu cầu và cung cấp thường tham khảo đến.

7.2 BIỂU DIỄN NGỮ NGHĨA CỦA DỊCH VỤ WEB

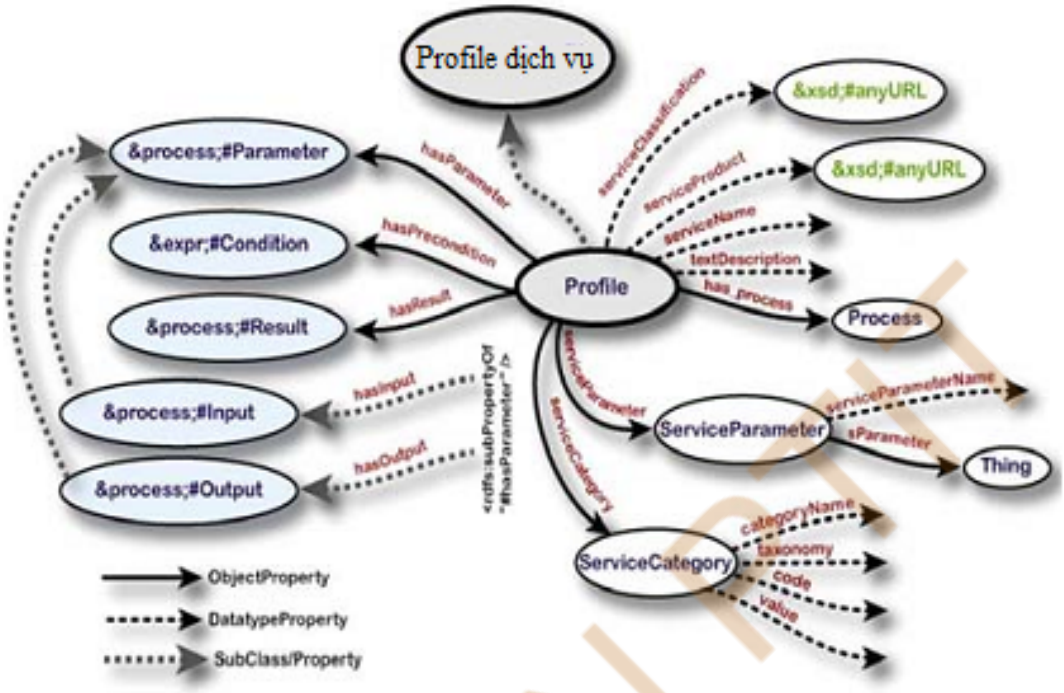
OWL-S là bản thể mức cao (upper ontology) được sử dụng để mô tả ngữ nghĩa của các dịch vụ dựa trên tiêu chuẩn W3C, bản thể OWL được tiếp nối từ WSDL. OWL-S có gốc gác từ bản thể dịch vụ DAML (DARPA Agent Markup Language) gọi là DAML-S ra đời từ năm 2001, và trở thành bộ phận của W3C năm 2005. OWL-S được xây dựng trên nền OWL và nó tổ chức biểu diễn dịch vụ bằng 3 bản thể là: hồ sơ (profile), mô hình tiến trình (process model) và nền tảng (grounding). Chi tiết về các bản thể này sẽ được trình bầy trong mục kế tiếp (6.3).

7.3 CÁC KHỐI XÂY DỰNG OWL-S

7.3.1 Bản thể OWL-S profile

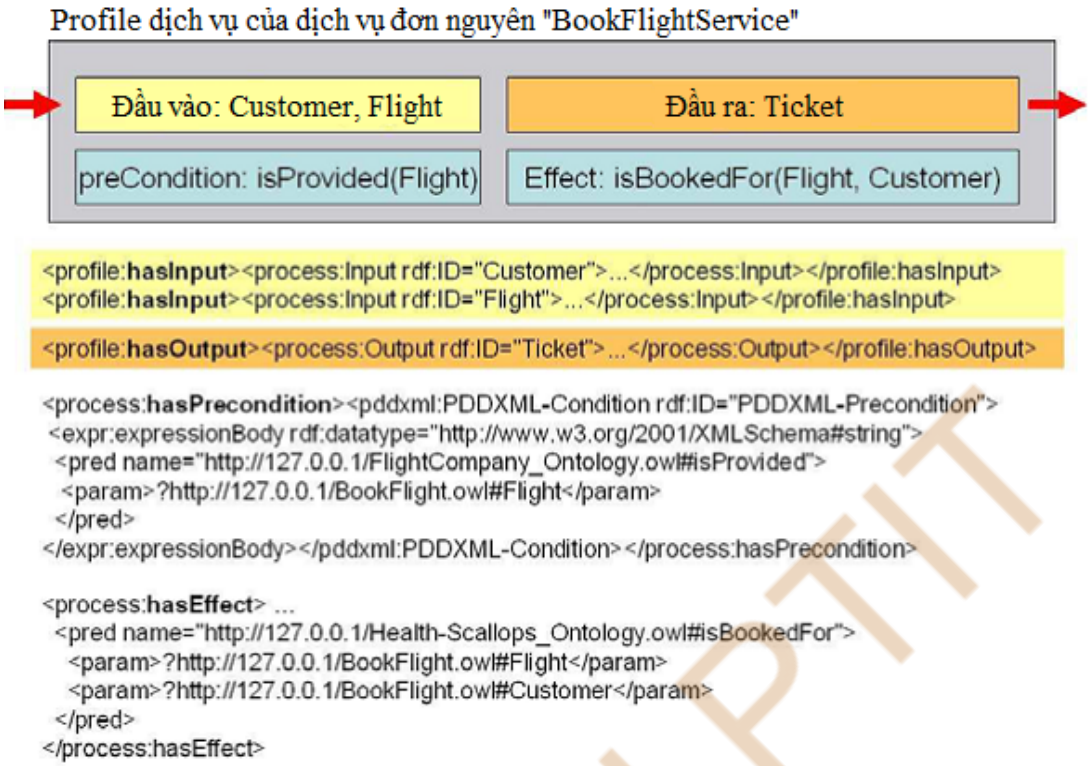
Bản thể OWL-S profile được sử dụng để mô tả dịch vụ làm gì và điều đó có ý nghĩa như thế nào cho mục đích khai phá dịch vụ. Bản thể OWL-S profile bao gồm chữ ký và các tham số chức năng của nó như hasInput, hasOutput, precondition và effect (IOPEs), hoặc các tham số phi chức năng như serviceName, serviceQuality, qualityRating, textDescription và siêu dữ liệu (meta-data) về nhà cung cấp dịch vụ và những yêu cầu dịch vụ. Chú ý rằng, khác với phiên bản OWL-S 1.0, trong phiên bản OWL-S 1.1 thì các tham số IOPE được định nghĩa

trong mô hình tiến trình (process model) với các tham khảo duy nhất cho các định nghĩa này từ profile. Điều này được minh họa trong hình 6.1.



Hình 6.1: Cấu trúc bản thể OWL-S profile [2]

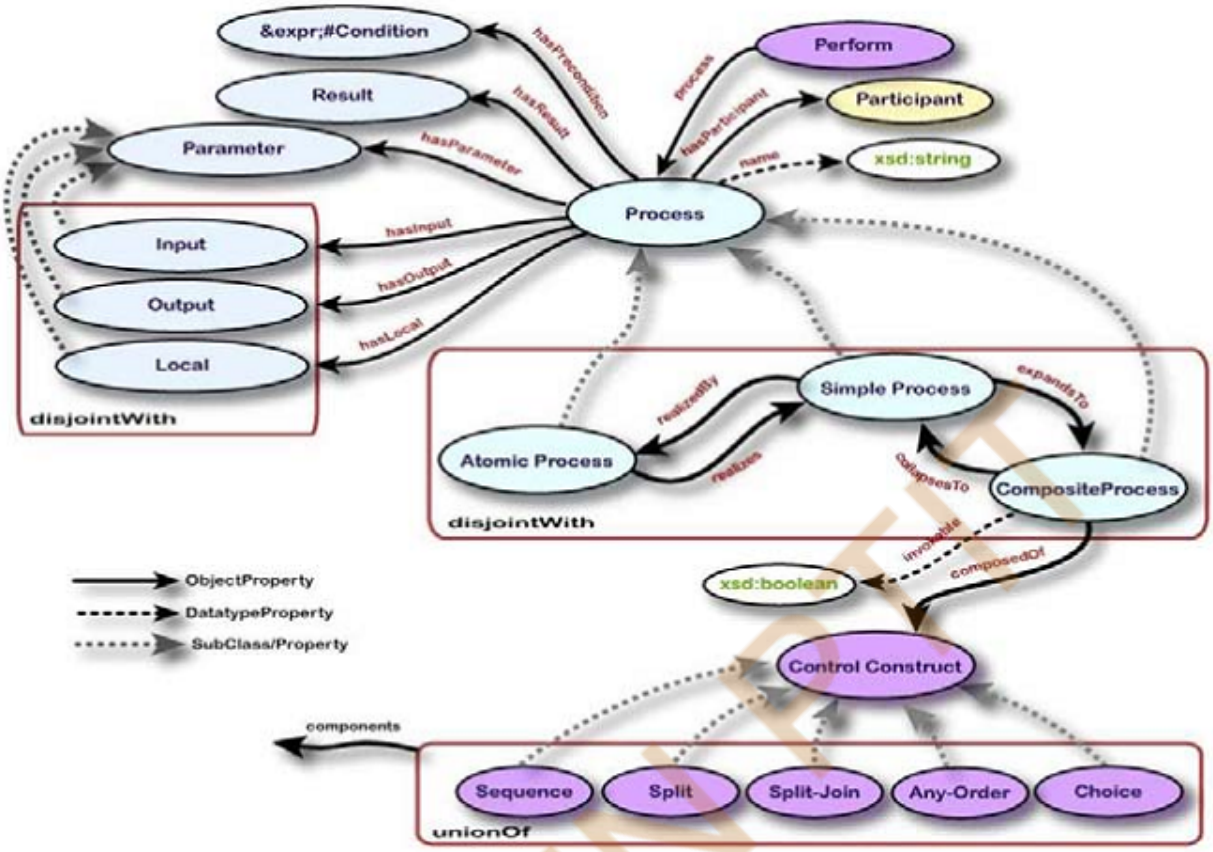
Đầu vào và đầu ra liên quan đến các kênh dữ liệu mà các luồng dữ liệu được sử dụng bởi các tiến trình. Preconditions (nguyên nhân) mô tả trạng thái của thế giới thực mà cần thiết cho tác tử khi thao tác với dịch vụ. Effects (kết quả) mô tả trạng thái kết quả sau khi tác tử thao tác lên dịch vụ. Ngược lại ngữ nghĩa của mỗi tham số đầu vào và đầu ra được định nghĩa như một khái niệm hình thức trong OWL được mô tả trong bản thể, điển hình là OWL-DL hoặc OWL-Lite. Preconditions và effects có thể được diễn tả bởi ngôn ngữ lô gics như KIF, PDDL và SWRL. Bên cạnh đó lớp profile có thể được chia thành các lớp con và được đặc tả, do đó hỗ trợ cho tạo ra các phân loại (taxonomies) mà nó mô tả các lớp khác nhau của dịch vụ. Ví dụ về dịch vụ web ngữ nghĩa profile sử dụng OWL-S 1.1 được trình bầy trong hình 6.2.



Hình 6.2: Dịch vụ web ngữ nghĩa profile sử dụng OWL-S 1.1 [2]

7.3.2 Bản thể OWL-S process

Bản thể OWL-S process mô tả sự hợp nhất của một hay nhiều dịch vụ. Trong OWL-S thì bản thể này được duy trì bởi một tập con chung các đặc trưng luồng công việc như split+join, tuần tự (sequence), và lựa chọn (choice) v.v. như mô tả trong hình 6.3. Lúc đầu process model được phát triển không phải cho mục đích khai phá dịch vụ mà cho việc liên kết OWL-S profile.

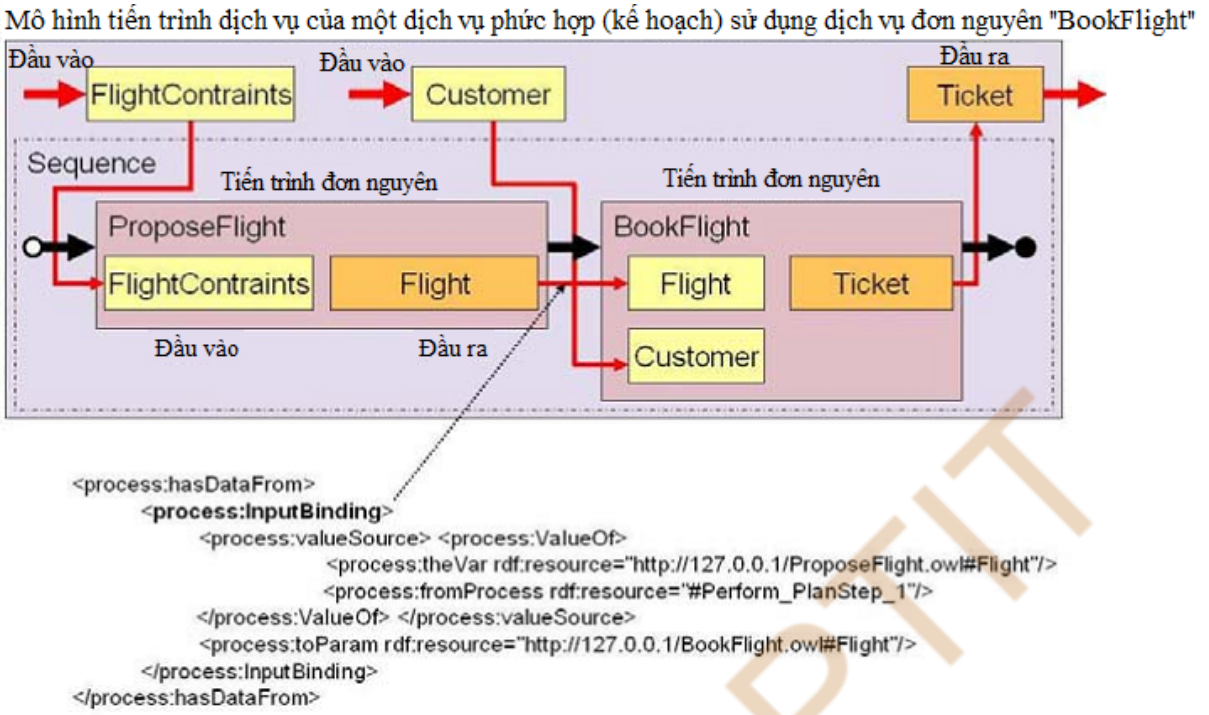


Hình 6.3: Mô hình tiến trình dịch vụ OWL-S [2]

Một cách chi tiết, một process (tiến trình) trong OWL-S có thể là đơn nguyên (atomic), hoặc hợp nhất (composite). Một tiến trình đơn nguyên là một tiến trình được mô tả như hộp đen với IOPE. Tiến trình này cung cấp dịch vụ mô tả hoặc trừu tượng hóa tiến trình phải nhận dạng bởi một tiến trình đơn nguyên. Ví dụ khai phá dịch vụ và ràng buộc động trong thời gian chạy hay việc mở rộng một tiến trình hợp nhất. Ví dụ về mô hình tiến trình với OWL-S được minh họa trong hình 6.4.

Các tiến trình hợp nhất (composite processes) là cac luồng công việc kiểu phân cấp; bao gồm các tiến trình đơn nguyên, đơn giản và các tiến trình phức hợp khác. Luồng công việc của các tiến trình này được xây dựng sử dụng một số phép toán điều khiển luồng như tuần tự (sequence), theo tác trên danh sách không theo thứ tự, lựa chọn (choice), if-then-else, lặp, repeat-while, split, split+join v.v.. Trong OWL-S 1.1, mô hình tiến trình cũng mô tả các đầu vào, đầu ra, preconditions, effects của tất cả các tiến trình mà là các phần của một dịch vụ hợp nhất, nó được tham chiếu từ profile của các dịch vụ tương ứng. Một mô hình tiến trình OWL-S của một dịch vụ phức hợp có thể đặc tả đầu ra của nó tương ứng với các đầu ra của một trong những tiến trình con. Hơn nữa, đối với một tiến trình phức hợp với một điều khiển tuần tự, đầu ra của một tiến trình con có thể là đầu vào của tiến trình con khác (như một ràng buộc).

Không may là ngữ nghĩa của mô hình OWL-S process lại không được định nghĩa trong các tài liệu chính thức của OWL-S. Do vây, đây là đề xuất cho những ngữ nghĩa này là các nghiên cứu về giải tích và lập trình logic.

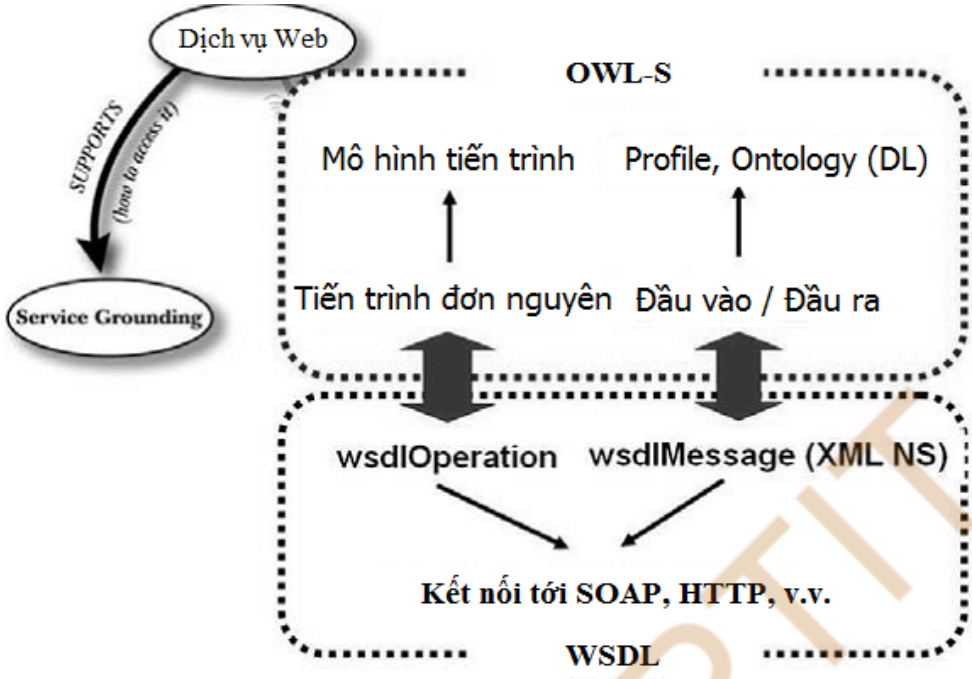


Hình 6.4: Ví dụ về mô hình tiến trình dịch vụ OWL-S [2]

7.3.3 Bản thể OWL-S grounding

Bản thể OWL-S grounding thực chất cung cấp ràng buộc giữa các định nghĩa dựa trên lô gic và XMLS cho mục đích dễ dàng thực thi dịch vụ. Bản thể OWL-S grounding có thể là một grounding trong WSDL kết nối đến OWL-S tới một chuẩn dịch vụ Web như hình 6.5 dưới đây.

Mô hình OWL-S process của một dịch vụ được ánh xạ tới biểu diễn WSDL qua một grouding đơn giản: Mỗi tiến trình đơn nguyên lại được ánh xạ tới một phép toán WSDL, và các thuộc tính OWL-S được sử dụng để mô tả đầu vào và đầu ra của các thông điệp vào, ra XML tương ứng. Không giống như OWL-S, WSDL không được sử dụng để diễn tả preconditions và effects trong khi thực thi dịch vụ. Dịch vụ đơn nguyên hay phức hợp với một grounding trong WSDL có thể được thực thi bằng lời gọi (invoke) của chương trình hoặc dịch vụ được tham chiếu trong file WSDL, hay thậm chí trong ngôn ngữ thực thi tiến trình kinh doanh (BPEL).



Hình 6.5: OWL-S grounding trong WSDL [2]

7.4 CÔNG CỤ PROTÉGÉ CHO XÂY DỰNG OWL-S

OWL-S là một plugin của công cụ Protégé. Như đã trình bầy trong mục 5.4 đây là phần mềm được phát triển bằng Java bởi các nhà nghiên cứu từ đại học Stanford. Do đó, trước khi cài đặt plugin OWL-S, ta cần tiến hành các bước sau:

- Cài đặt máy ảo Java (JVM),
- Download và cài đặt Protégé từ trang web:

  http://www.protege.stanford.edu/download.html
- Download plug-in OWL từ địa chỉ:

  http://www.protege.stanford.edu/plugins/owl/download.html
- Giải nén vào thư mục < Protégé-dir >/plugins.
- Download và cài đặt Graphviz từ địa chỉ:

  http://www.research/att/com/sw/tools/graphviz/download.html

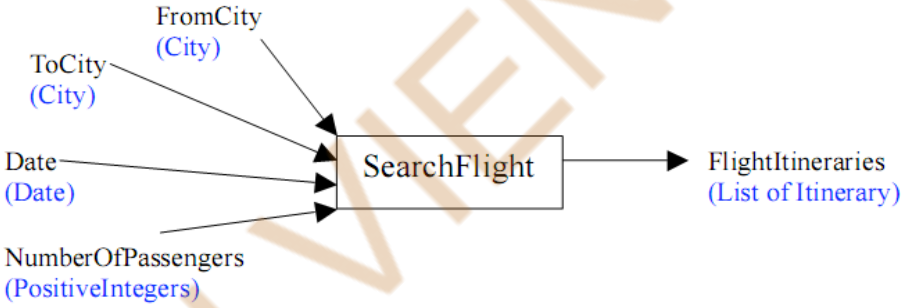
  Phần mềm này cần thiết để chạy hệ soạn thảo OWL-S với giao diện đồ họa.

Sau khi hoàn thành các bước trên ta download OWL-S plugin từ địa chỉ http://owlseditor.semwebcentral.org sau đó giải nén vào thư mục < Protégé-dir >/plugins. Sau khi hoàn thiện cài đặt thì trong thư mục Protégé có cấu trúc như hình 6.6.

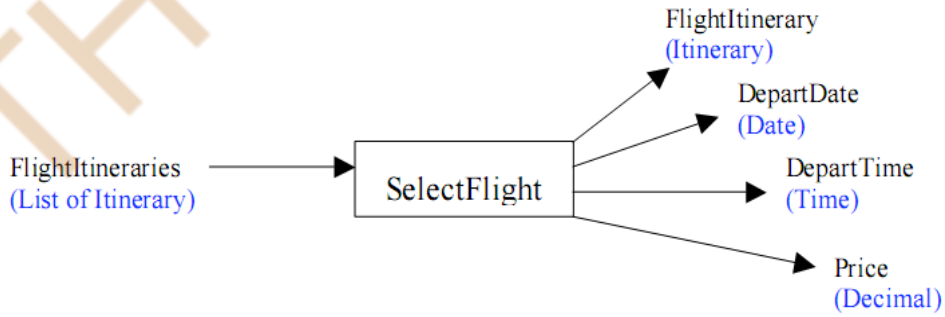
Hình 6.6: Hoàn thiện cài đặt bộ soạn thảo OWL-S

Đến đây bạn có thể các project theo định dạng khác nhau chẳng hạn owl hoặc rdf. Phần dưới đây là hướng dẫn xây dựng một dịch vụ OWL-S sử dụng plugin OWL-S.

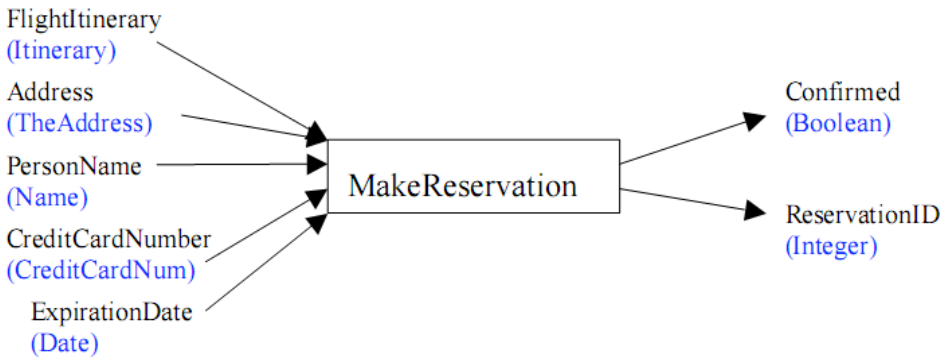
Giả sử ta muốn xây dựng một dịch vụ web tên là BravoAir [4] cung cấp các chức năng: SearchFlight (tìm kiếm chuyến bay), SelectFlight (lựa chọn chuyến bay mong muốn), và MakeReservation (đặt chỗ). Tương ứng với 3 chức năng này là 3 tiến trình đơn nguyên (atomic process). Trước hết ta cần xác định đầu vào và đầu ra cho các tiến trình đơn nguyên (atomic process) này. Các lược đồ dưới đây minh họa điều đó:



Hình 6.7: Đầu vào và đầu ra của tiến trình đơn nguyên SearchFlight

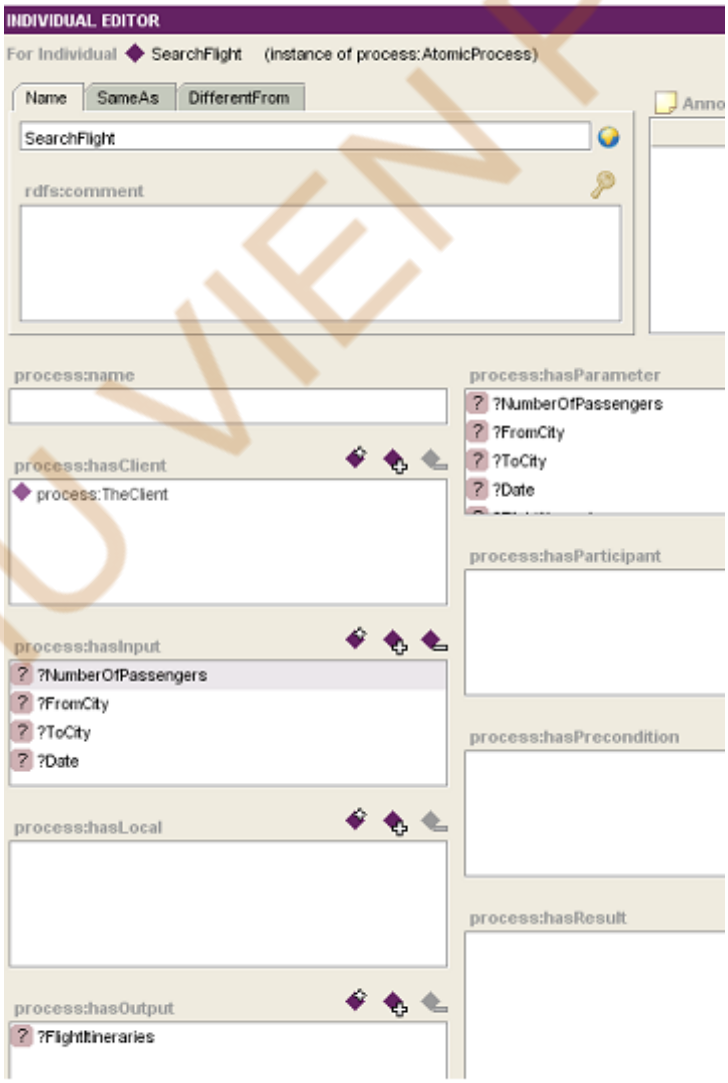


Hình 6.8: Đầu vào và đầu ra của tiến trình đơn nguyên SearchFlight



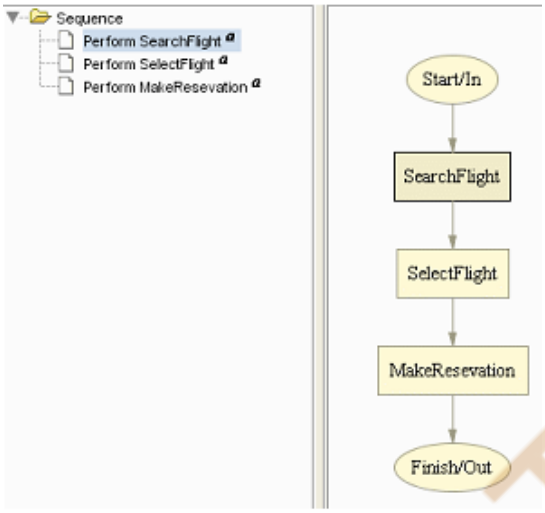
Hình 6.9: Đầu vào và đầu ra của tiến trình đơn nguyên SearchFlight

Tương ứng với đầu vào và đầu ra của tiến trình đơn nguyên ta xác định các kiểu lớp (type of class), kiểu tên và tham số (name and parameter type) cho từng tham số đầu vào và ra của mỗi tiến trình đơn nguyên; hình 6.10 minh họa các tham số của tiến trình đơn nguyên SearchFlight.



Hình 6.10: Minh họa các tham số của tiến trình đơn nguyên SearchFlight

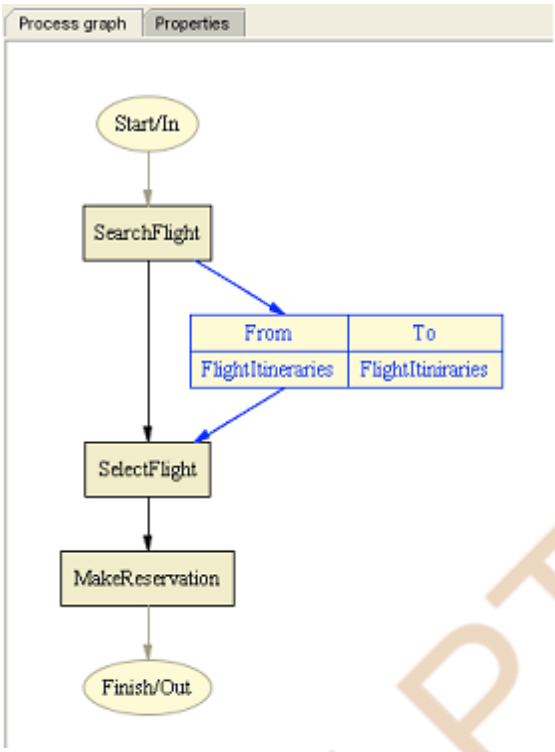
Bước tiếp theo tạo ra đồ thị biểu diễn mối liên hệ giữa các tiến trình. Trong trường hợp này, dịch vụ cần tạo khá đơn giản nên đồ thị có thể được hiển thị dạng cây như hình 6.11.



Hình 6.11: Đồ thị của các tiến trình

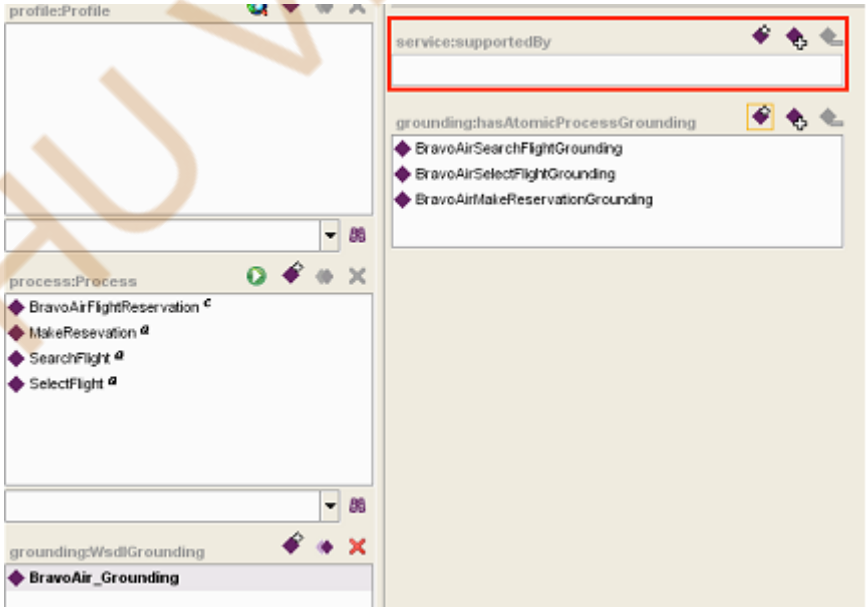
Tiếp theo ta định nghĩa luồng dữ liệu. Trong trường hợp này đầu ra của tiến trình SearchFlight là FlightItineraries (các hành trình của chuyến bay) có thể là đầu vào của tiến trình SelectFlight. Trong cac trường hợp khác ví dụ như tiến trình là một tiến trình phức hợp thì luồng dữ liệu cần đươc xác định cho các tiến trình con.

Để xây dựng luồng dữ liệu thì các tham số cần được rang buộc (binding) giá trị dữ liệu, giá trị nguồn (dữ liệu đi từ đâu) và giá trị hàm. Ta phải thêm các câu lệnh SearchFlightPerform hay SelectFlightPerform để tạo giới hạn cho khoảng giá trị của tham số đầu vào. Bộ soạn thảo OWL-S sẽ tự động tạo ra một biểu diễn luồng dữ liệu trên đồ thị tiến trình. Để vẽ lại đồ thị tiến trình, các nút trên đồ thị có thể phải được khóa lại. các đồ thị con với các nút mầu vàng và chữ mầu xanh thể hiện luồng dữ liệu. Hình 6.12 là luồng dữ liệu cho dịch vụ BravoAir.



Hình 6.12: Luồng điều khiển và dữ liệu

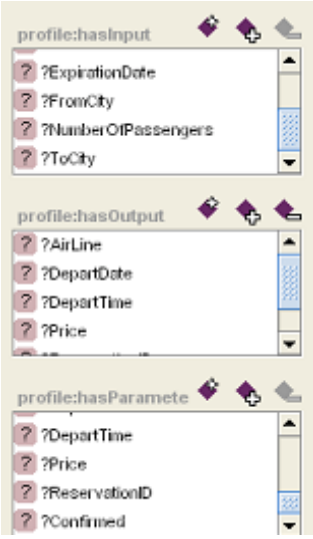
Bước tiếp đến là tạo OWL-grounding cho dịnh vụ. Trước hết ta cần tạo grounding có tên BravoAirSearchFlightGrounding cho tiến trình SearchFlight, thì biến thể hiện của grounding sẽ là grounding:WSDLAtomicProcessGrounding. Tương tự như vậy đối với các tiến trình SelectFlight và MakeReservation.



Hình 6.13: groundings cho các tiến trình của dịch vụ BravoAir

Bước tiếp theo là tạo OWL-S profile cho dịnh vụ. Đầu tiên ta cần lựa chọn các tham số cho profile thường là tập con của tất cả các tham số đầu vào và đầu ra của các tiến trình. Việc này quyết định bởi người thiết kế dịch vụ sẽ lựa chọn những tham số nào được xuất hiện trong profile của dịch vụ. Trong ví dụ này ta sẽ sử dụng tất cả các tham số đã được định nghĩa ngoại

trừ các tham số nội bộ như FlightItineraries. Kết quả các tham số trong profile được trình bầy như hình 6.14.



Hình 6.14: Các tham số profile của dịch vụ

Bước cuối cùng là tạo dịch vụ và đặt tên cho nó. Sau đó liên kết dịch vụ vừa tạo với profile, các tiến trình và grounding mà ta đã tạo ở các bước trước. Ta cũng cần phải gán các biến thể hiện với các thuộc tính thích hợp. Chẳng hạn, biến thể hiện BravoAir_Profile cần có thuộc tính service:presents, BravoReservationFlight được gán cho service:describedBy, hay BravoAir_Grounding được gán cho service:supports.

7.5 BÀI TẬP

1. Hãy xây một dịch vụ Web sử dụng công cụ Protégé và plug-in OWL-S để cung cấp dịch vụ tra cứu điểm các môn học trong một học kỳ cho sinh viên đại học. Cơ sở dữ liệu sinh viên cần được tạo trên hệ quản trị cơ sở dữ liệu do bạn lựa chọn (như SQL server, MS Access, MySQL v.v..). Cơ sở dữ liệu này có các bảng chứa các thông tin về sinh viên như: mã sinh viên, họ và tên sinh viên, học kỳ (học kỳ mấy?), danh sách các môn học theo từng học kỳ và điểm trung bình của từng môn học.

   Trong dịch vụ Web có một phương thức là Tracuu với 2 tham số đầu vào là mã sinh viên và học kỳ; phương thức này sẽ trả về tên sinh viên, danh sách các môn học và điểm của từng môn theo học kỳ.

   Học viên cũng được yêu cầu phát triển ứng dụng phía client để gọi đến (invoke) phương thức Tracuu và hiển thị thông tin trả về dưới dạng một bảng.

   Học viên được yêu cầu tuân thủ các bước như ví dụ quản lý đặt chỗ cho chuyến bay ở mục 6.4.

CHƯƠNG 8
KHÁM PHÁ DỊCH VỤ WEB NGỮ NGHĨA

8.1 KHÁM PHÁ DỊCH VỤ WEB NGỮ NGHĨA

Khám phá dịch vụ Web ngữ nghĩa là quá trình xác định vị trí mà các dịch vụ web đang tồn tại dựa vào biểu diễn các ngữ nghĩa chức năng hoặc phi chức năng của các dịnh vụ web. Tình huống khám khá thường xảy ra khi ta đang cố gắng sử dụng một chức năng (function) nào đó của một dịch vụ web trong lúc ta đang xây dựng một tiến trình kinh doanh mới hoặc nâng cấp các tiến trình kinh doanh hiện có. Cả hai cách tiếp cận tính toán hướng dịch vụ và dịch vụ web ngữ nghĩa đều có thể giaỉ quyết vấn đề này.

Khám phá dịch vụ có thể được thực hiện theo nhiều cách khác nhau tùy thuộc vào nền tảng biểu diễn dịch vụ, công cụ lựa chọn dịch vụ, và cách thức kết hợp với nhau. Nhìn chung, nền tảng khám phá dịch vụ cần có những thành phần sau:

- *Biểu diễn dịch vụ*: Các công cụ hình thức để mô tả các ngữ nghĩa chức năng hoặc phi chức năng của các dịnh vụ web.
- *Lựa chọn dịch vụ*: Các cơ chế suy diễn để đối sánh dịch vụ (service matching), đó là các cặp đối sánh dựa trên biểu diễn dịch vụ: truy vấn (query), xếp hạng (ranking), ộ tương đồng một phần hay toàn bộ của phép đối sánh.
- *Kiến trúc khám phá*: Bao gồm môi trường giả định trên một hình trạng mạng (network topology) như không tập trung (decentralized) hay tập trung (centralized), lưu trữ thông tin dịch vụ, và các cơ chế xác định vị trí, và các vai trò của tác tử (là yêu cầu dịch vụ, cung cấp dịch vụ hay là chuyển tiếp dịch vụ).

8.2 THIẾT KẾ CƠ CHẾ KHÁM PHÁ DỊCH VỤ WEB NGỮ NGHĨA

Đối sánh dịch vụ ngữ nghĩa xác định xem ngữ nghĩa của một dịch vụ có tuân thủ như đã được quảng bá (advertised) hay không. Đây chính là vấn đề cốt lõi của bất kỳ nền tảng khám phá dịch vụ Web ngữ nghĩa nào. Các cách tiếp cận đối sánh ngữ nghĩa của dịch vụ có thể phân làm hai loại theo:

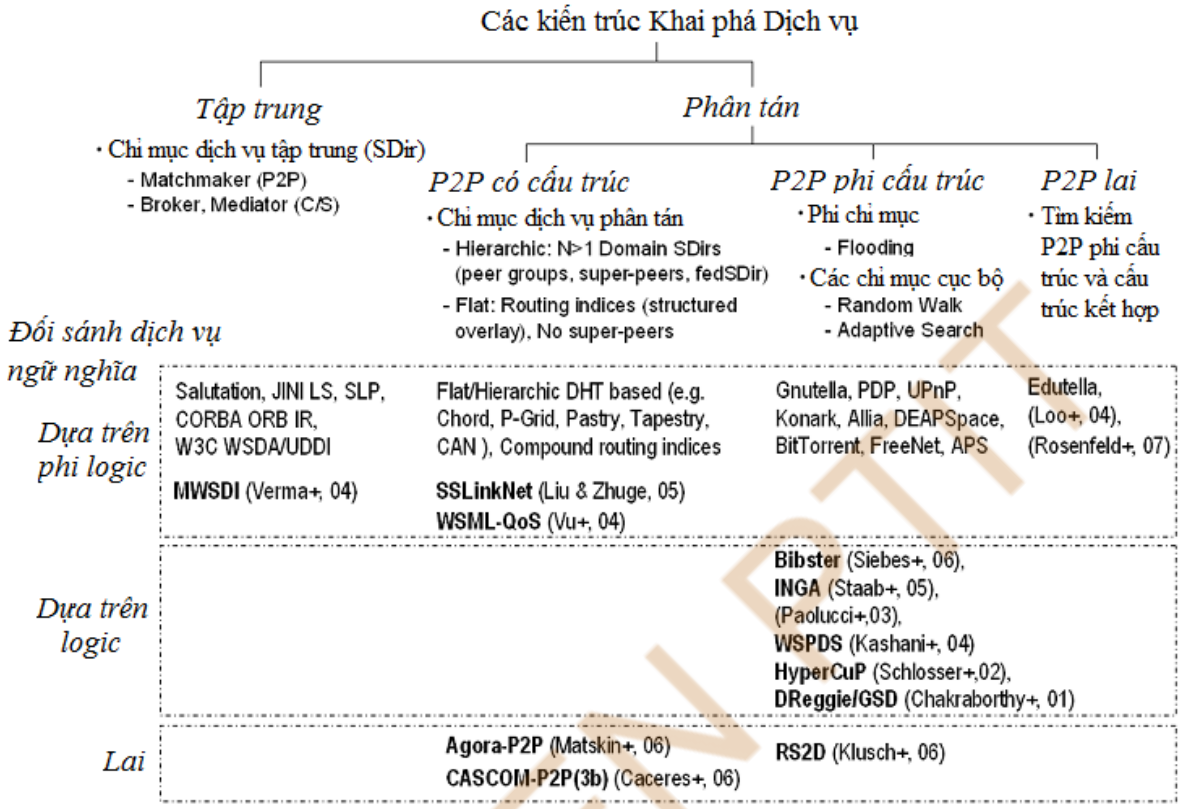
- Loại và những phần ngữ nghĩa của dịch vụ được xem xét đối sánh, và:
- Cách thức đối sánh được thực hiện như thế nào.

Trong phần này ta sẽ tìm hiểu kiến trúc của cơ chế khám phá ngữ nghĩa của dịch vụ và các thuật toán đối sánh ngữ nghĩa của dịch vụ.

8.2.1 Kiến trúc cơ chế khám phá

Các kiến trúc của cơ chế khám phá ngữ nghĩa của dịch vụ có thể được phân thành hai loại: tập trung và không tập trung như minh họa trong hình 7.1. Cách phân loại này dựa trên cách thức lưu trữ thông tin và xác định vị trí của ngữ nghĩa của dịch vụ. Các hệ thống khám phá dịch vụ tập trung dựa trên một dịch vụ danh bạ toàn cục duy nhất (nhưng có thể thay thế) và được duy trì bởi một tác tử như tác tử đối sánh (matchmaker), môi giới hoặc hòa giải.

Ngược lại các hệ thống khám phá dịch vụ không tập trung dựa trên sự lưu trữ thông tin phân tán trên nhiều dạng mạng ngang hàng: mạng có cấu trúc, mạng phi cấu trúc hoặc mạng lai.



Hình 7.1: Kiến trúc tập trung và không tập trung của cơ chế khám phá ngữ nghĩa của dịch vụ [2]

Các hệ thống khám phá ngữ nghĩa của dịch vụ dựa trên kiến trúc tập trung:

Trong các hệ thống khám phá ngữ nghĩa của dịch vụ dựa trên kiến trúc tập trung, một danh bạ dịch vụ tập trung hoặc một tác tử đối sánh trả về một danh sách các nhà cung cấp của những dịch vụ tương ứng về ngữ nghĩa cho người yêu cầu. Ngược lại với cơ chế xử lý tập trung client-server của phần mềm trung gian hoặc môi giới, người yêu cầu dịch vụ có thể tương tác trực tiếp với những nhà cung cấp được lựa chọn cho việc cung cấp dịch vụ. Ưu điểm của danh bạ dựa trên kiến trúc tập trung là quản lý tài nguyên hiệu quả và thời gian tìm kiếm nhanh, mặc dù máy chủ tìm kiếm tập trung hay đăng kí như JINI hoặc giao diện đăng kí CORBA ORB là một điểm duy nhất.

Một ứng dụng của danh bạ dịch vụ tập trung là hệ thống chia sẻ file âm nhạc Napster, và hệ thống SETI@home tận dụng tài nguyên phân tán khổng lồ trên mạng cho việc xử lý tìm kiếm. Theo cách nhìn của khám phá ngữ nghĩa của dịch vụ Web, mỗi một tác tử đối sánh ngữ nghĩa của dịch vụ web tự nó sẽ là một hệ thống ngữ nghĩa của dịch vụ web theo kiến trúc tập trung. Ví dụ như hệ thống dịch vụ chăm sóc sức khỏe điện tử SCALLOPS e-health sử dụng tác tử đối sánh OWLS-MX như một dịch vụ tập trung để lựa chọn y tế điện tử phù hợp cho những trường hợp cấp cứu.

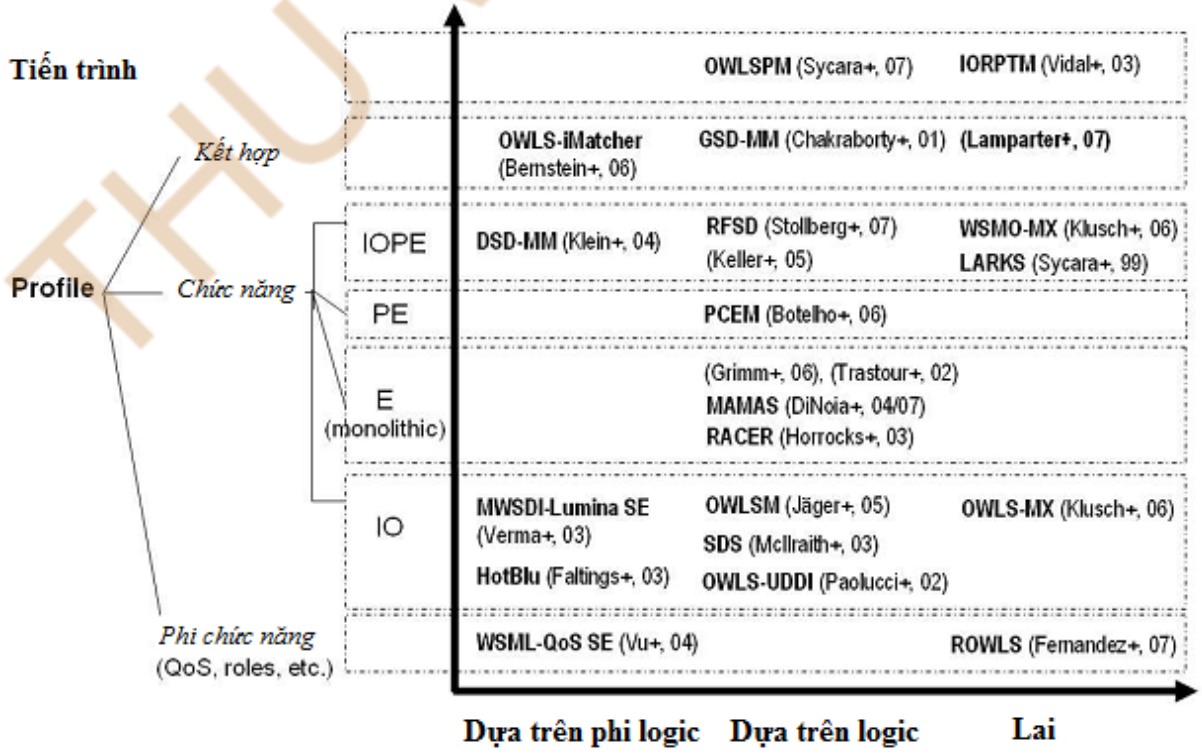
Các hệ thống khám phá ngữ nghĩa dịch vụ dựa trên kiến trúc không tập trung:

Các hệ thống khám phá ngữ nghĩa dịch vụ dựa trên kiến trúc không tập trung dựa trên các cơ chế lưu trữ thông tin và xác định vị trí trên các mạng ngang hàng. Các mạng ngang hàng này có thể được chia thành 3 loại: có cấu trúc, phi cấu trúc và mạng lai.

Các hệ thống khám phá ngữ nghĩa dịch vụ dựa trên kiến trúc không tập trung dựa trên mạng ngang hàng có cấu trúc không có máy chủ danh bạ trung tâm nhưng lại có một số lượng lớn hình trạng mạng (topology được điều khiển một cách chặt chẽ. Các tài nguyên không được đặt một cách ngẫu nhiên hay trong danh bạ trung tâm mà được phân phát theo nhu cầu xử lý các yêu cầu từ người dùng. Nói cách khác các chỉ số dịch vụ của hệ thống được phân tán tới các mạng ngang hàng theo một cách trải dàn (overlay) có cấu trúc phụ thuộc vào nội dung xác định được phân tán và nó có thể được sử dụng cho định tuyến tới các điểm có yêu cầu (point queries). Ngược lại, trong các hệ thống khám phá ngữ nghĩa dịch vụ dựa trên kiến trúc không tập trung dựa trên mạng ngang hàng phi cấu trúc lại không có chỉ số hay việc điều khiển chính xác trên hình trạng mạng mà các file lưu trữ dựa trên tri thức của hình trạng mạng. Có nghĩa là việc định tuyến các tài nguyên theo yêu cầu không bị hạn chế bởi danh bạ dịch vụ mà chúng thực hiện. Các hệ thống khám phá ngữ nghĩa dịch vụ dựa trên kiến trúc không tập trung dựa trên mạng ngang hàng lai (hybrid P2P) kết hợp cả hai cơ chế xác định vị trí có cấu trúc và không có cấu trúc. Nghĩa là cùng với việc định tuyến chỉ số dịch vụ và phát tán hiệu quả các tài nguyên trong mạng.

8.2.2 Các thuật toán đối sánh

Dưới đây là một số thuật toán đối sánh ngữ nghĩa của dịch vụ (semantic service matching); Phân loại các thuật toán đối sánh ngữ nghĩa dịch vụ được trình bầy trong hình 7.2. Trong hình vẽ trên các thuật toán đối sánh có thê được chia thành hai nhóm: nhóm thứ nhất là: đối sánh ngữ nghĩa của dịch vụ dựa trên phi lo gic (non-logic), logic và lai (hybrid); nhóm thứ hai là đối sánh ngữ nghĩa của dịch vụ dựa trên profile và mô hình process.



Hình 7.2: Phân loại các thuật toán đối sánh ngữ nghĩa dịch vụ [2]

Đối sánh ngữ nghĩa của dịch vụ dựa trên lô gic:

Đối sánh ngữ nghĩa của dịch vụ dựa trên logic thực hiện các phép suy diễn logic trên ngữ nghĩa của dịch vụ. Thuật toán này tập trung vào so sánh ngữ nghĩa hình thức theo từng cặp dịch vụ. Các khái niệm lo gic hay luật suy diễn được sử dụng để định nghĩa ngữ nghĩa này được đặc tả trong các bản thể được coi như lý thuyết cơ bản như suy diễn cấp một (first-order) hoặc suy diễn dựa luật (rule-base) trên cùng một tập từ vựng tối thiểu. Các bản thể khác nhau của các nhà cung cấp dịch vụ và người yêu cầu dịch vụ phải đối sánh trong khoảng thời gian thiết kế, hay trong thời gian chạy như một phần có quá trình đối sánh. Các cấp độ đối sánh (matching degrees) của một cặp dịch vụ ngữ nghĩa được xác định bằng cách suy diễn logic hoặc kết hợp suy diễn lo gic với việc thực hiện các thuật toán khác (không có trong suy diễn lo gic). Các cấp độ đối sánh thường dùng là chính xác (exact), plugin, gộp vào (subsume), và phép tách (disjoin) mà được xác định khác nhau dựa vào các phần ngữ nghĩa của dịch vụ là kiểu lý thuyết logic sử dụng để tính toán chúng.

Đối sánh ngữ nghĩa của dịch vụ dựa trên phi lô gic:

Như đã trình bầy ở trên, đối sánh ngữ nghĩa của dịch vụ dựa trên phi lô gic không thực hiện bất kỳ phép suy diễn logic nào trên ngữ nghĩa của các dịch vụ. Thay vào đó, chúng tính độ đối sánh ngữ nghĩa với cặp biểu diễn dịch vụ cho trước. Chẳng hạn, để đo độ tương tự của cú pháp, đối sánh đồ thị có cấu trúc, hay các phép tính khoảng cách các khái niệm dạng số trên các bản thể cho trước. Có nhiều độ đo tương tự giữa hai văn bản từ lĩnh vực nghiên cứu trích xuất thông tin (information retrieval), khai phá mẫu xấp xỉ, hay gom nhóm dữ liệu (data clustering) từ data mining, hay xếp hạng từ khóa, tìm kiếm tài liệu XML có cấu trúc với XQuery v.v.. Nói chung đối sánh ngữ nghĩa của dịch vụ dựa trên phi lô gic tận dụng ngữ nghĩa ngầm định (implicit) bên trong. Ví dụ như các từ mẫu (pattern), đố thị con (subgraph), hay tần suất tương đối (relative frequency) được sử dụng nhiều trong biểu diễn dịch vụ, hơn là các khai báo IOPE được mô tả rõ ràng (explicitly) trong lo gic.

Đối sánh ngữ nghĩa profile của dịch vụ dựa trên phương pháp lai (Hybrid Semantic Profile Matching)

Các kỹ thuật đối sánh cú pháp là mục tiêu đầu tiên cho việc phát triển các giải pháp đối sánh ngữ nghĩa profile của dịch vụ dựa trên phương pháp lai kết hợp giữa đối sánh dựa trên lo gic và phi lo gic vì nếu đối sánh chỉ dựa trên lo gic hoặc phi lo gic đều không đáp ứng được yêu cầu này. Nhiều thực nghiệm đánh giá rằng phương pháp đối sánh lai cho kết quả tốt hơn nhiều so với phương pháp đối sánh chỉ dựa trên lo gic hoặc phi lo gic dưới cùng điều kiện.

Đối sánh ngữ nghĩa mô hình process của dịch vụ dựa trên lô gic:

Đối sánh ngữ nghĩa mô hình process của dịch vụ nhìn chung là không thông dụng và không là chủ ý của các nhà thiết kế của biểu diễn ngữ nghĩa của dịch vụ Web. Bên cạnh đó ngữ nghĩa của mô hình process trong OWL-S hay WSDL chưa được định nghĩa một cách hình thức. Đây là vấn đề mà có thể được giải quyết một phần bằng cách viết lại các biểu diễn mô hình process theo lô gic thích hợp với hệ thống tự động chứng minh với các công cụ hỗ trợ phân tích.

Đối sánh ngữ nghĩa mô hình process của dịch vụ dựa trên phi lô gic và lai:

Đối sánh ngữ nghĩa mô hình process của dịch vụ dựa trên phi lô gic có thể thích hợp được áp dụng để biến đổi các cặp ngữ nghĩa mô hình process của dịch vụ web. Chẳng hạn, cách tiếp cận để đối sánh đồ thị phụ thuộc tiến trình dựa trên độ đo tương tự cú pháp bằng phép lai. Phép đệ qui so sánh đồ thị phụ thuộc mô hình tiến trình dựa trên các nhiệm vụ của luồng công việc và đối sánh lo gic giữa các tham số vào/ra của các nút trong đồ thị biểu diễn tiến trình các dịch vụ con. Nói cách khác, đối sánh tiến trình dịch vụ theo chức năng có thể được tận dụng để tìm kiếm một tập các dịch vụ con của một dịch vụ phức hợp.

8.2.3 Chi tiết cài đặt

Trong mục này giới thiệu một trong những crawler được thiết kế để rút trích nội dung web ngữ nghĩa đó là Slug. Slug là crawler mã nguồn mở Java có thể được download tại địa chỉ: http://www.ldodds.com/projects/slug/

Slug cho phép người dùng có thể đặt cấu hình một cách mềm dẻo cho các thao tác: rút trích, xử lý và lưu trữ nội dung của dịch vụ web ngữ nghĩa bằng cách sửa các tham số trong file cấu hình. Hơn nữa, slug cung cấp từ vựng RDF để mô tả các cấu hình crawler và thu thập siêu dữ liệu liên quan đến các hoạt động crawling. Các siêu dữ liệu crawler cho phép báo cáo và phân tích quá trình crawling và việc trích rút thông tin hiệu quả hơn thông qua lưu trữ vào vùng đệm (caching) của HTTP.

a. Cài đặt crawler mô tả dịch vụ web ngữ nghĩa

Trước hết cần download địa chỉ trên (phiên bản alpha-2); sau đó tiến hành cài đặt. Giải nén trong thư mục home. Thư mục con \slug chứa tất cả các file code và các file cấu hình. Bạn cũng nên tạo thư mục và biến môi trường $SLUG_HOME để tham chiếu đến slug và cập nhật đường dẫn này vào biến PATH. JVM và Java 1.5 trở nên cần được cài trước. Sau đó, bạn cần cài Apache Ant để build code. Bước cuối cùng là thiết lập cấu hình trong file config.

b. Cài đặt kho chứa mô tả dịch vụ web ngữ nghĩa

Việc cài đặt kho chứa mô tả dịch vụ web trên slug ta có thể thiết lập từ file cấu hình, cụ thể là ta có thể edit nội dung đoạn mã XML trong file cấu hình (hình 7.3). Bên cạnh đó ta có thể thiết lập các thông tin sau:

- Số luồng có thể chạy đồng thời trong khi trích rút dữ liệu
- Vị trí đặt kho chứa mô tả dịch vụ web,
- Thành phần nào sẽ xử lý rút trích dữ liệu
- Crawler sử dụng các bộ lọc nào để tìm các URL mới

```
  <slug:Scutter rdf:about="persistent-scutter">
          <dc:description>A Scutter that includes writing incoming data into
          a persistent memory. Note that the memory is different to that
          holdings Scutter persistent state.</dc:description
<slug:hasMemory rdf:resource="memory"/>
 <!-- how many worker threads? -->
    <slug:workers>10</slug:workers>
    <!-- configures consumers for incoming data. Added persistent-storer -->
          <slug:consumers>
            <rdf:Seq>
             <rdf:li rdf:resource="storer"/>
             <rdf:li rdf:resource="rdf-consumer"/>
             <rdf:li rdf:resource="persistent-storer"/>
            </rdf:Seq>
          </slug:consumers>
          <!-- configures filter pipeline for controller -->
          <slug:filters>
            <rdf:Seq>
              <rdf:li rdf:resource="single-fetch-filter"/>
              <rdf:li rdf:resource="depth-filter"/>
              <rdf:li rdf:resource="regex-filter"/>
            </rdf:Seq>
          </slug:filters>
  </slug:Scutter>
```

Hình 7.3: Cấu hình kho chứa mô tả dịch vụ web ngữ nghĩa

CHƯƠNG 9
LỰA CHỌN DỊCH VỤ WEB NGỮ NGHĨA

9.1 KHÁI NIỆM LỰA CHỌN DỊCH VỤ

Lựa chọn dịch vụ bắt đầu với khám phá dịch vụ. Để dịch vụ có thể được sử dụng thì nó cần phải được khám phá bởi một người dùng và cần có một trao đổi (correspondence) tương ứng được thiết lập các mục tiêu của người dung và khả năng đáp ứng của dịch vụ. Một ứng dụng không cần phải khám phá tất cả các dịch vụ đối sánh với một tập các yêu cầu, thậm chí với lời giải tối ưu nhất, nhưng cần tìm lời giải đủ tốt theo nghĩa đặc tính và chất lượng. Bằng việc dựa trên khám phá đáp ứng các yêu cầu đặc tả chất lượng dịch vụ (QoS). Ta có thể giảm các kết quả trung gian không thích hợp, và do đó có thể giảm chi phí tính toán cho việc khám phá. Lựa chọn dịch vụ có thế được tiếp cận từ 3 quan điểm:

- Người dùng tìm kiếm nhà cung cấp tiềm năng
- Nhà cung cấp tìm kiếm người dùng tiềm năng
- Môi giới tìm kiếm cả người dùng và nhà cung cấp trong một phiên đấu giá hay trong SoCom.

Cả 3 quan điểm trên có thể hưởng lợi từ việc biểu diễn ngữ nghĩa của chất lượng thuộc tính. Nhắc lại rằng các yêu cầu chức năng là những gì dịch vụ có thể làm (vi dụ: IOPE của OWL-S) và những yêu cầu phi chức năng là chất lượng của dịch vụ (độ tin cậy, tính sẵn sàng). Những biểu diễn tốt hơn dựa trên các bản thể có thể được sử dụng cho cả những yêu cầu chức năng hoặc phi chức năng.

Người dùng có thể nhận nhiều gợi ý đối với các dịch vụ thỏa mãn hoặc gần thỏa mãn yêu cầu của họ thì họ cần phải lựa chọn dịch vụ thích hợp giữa nhiều dịch vụ khác. Việc này cần sử dụng thêm nhiều tiêu chuẩn khác nữa, ví dụ: lựa chọn dịch vụ in ấn giá rẻ nhất hay cần thêm thông tin của bên thứ ba như danh tiếng của dịch vụ in. Hay thậm chí thương lượng với các dịch vụ để xem xét dịch vụ nào phù hợp nhất thỏa mãn yêu cầu của mình.

9.2 LỰA CHỌN DỰA TRÊN ĐỐI SÁNH NGỮ NGHĨA

Khám phá ngữ nghĩa của dịch vụ có thể khả thi bằng một tập các biểu diễn ngữ nghĩa, tương tự như việc quảng bá trong thế giới vật chất. Các biểu diễn WSDL có thể được dùng cho việc quảng bá này (ví dụ: cho phép thực hiện các lời gọi trong một phiên bản demo của dịch vụ web), nhưng thiếu ngữ nghĩa cho các thuộc tính chức năng và không đặc tả các thuộc tính phi chức năng. Các quảng cáo trong thực tế cuộc sống (như cung cấp một dịch vụ điện thoại đường dài, giá rẻ) nếu được diễn tả trong ngôn ngữ lo gic hình thức sẽ là những ví dụ thú vị của biểu diễn ngữ nghĩa những khả năng của dịch vụ.

Các biểu diễn ngữ nghĩa có thể được tạo ra bởi nhà cung cấp dịch vụ hoặc bên thứ ba. Nhưng bên thứ ba thường tạo ra các biểu diễn dịch vụ mà không cung cấp những biểu diễn này. Chẳng hạn, có nhiều trang web về khách sạn, nhưng hầu hết các biểu diễn ngữ nghĩa và nội dung của những dịch vụ web này đều không được cung cấp trên internet. Một đại lý dịch vụ du lịch có thể chọn để cung cấp các biểu diễn của những trang web này mà các khách hàng

của đại lý đó có thể sử dụng. Các biểu diễn này có thể được tạo ra theo yêu cầu của dịch vụ. Chúng có thể được bảo trì trên máy tính cục bộ, được lưu giữ bởi bên thứ ba, gửi cho các tác tử đối sánh trung gian, hoặc lưu trữ tại một điểm đăng ký tập trung. Điều này cũng hàm ý rằng đối sánh ngữ nghĩa càng phức tạp thì việc biểu diễn ngữ nghĩa càng phong phú hơn (richer). Nói cách khác, các biểu diễn phong phú cho phép lựa chọn chính xác hơn.

Về phía khách hàng, những mục tiêu của đại lý phải được mô tả trong một ngôn ngữ hình thức. Mục đích của việc đối sánh là tìm một dịch vụ đủ tốt tương tự giữa mục tiêu của người dùng và khả năng đáp ứng của dịch vụ mà đã quảng cáo. Nhìn chung, đối sánh thường được xác định bởi các thuật toán heuristic với các bản thể đã đặc tả các thuật ngữ được sử dụng trong quảng cáo và biểu diễn các mục tiêu của đại lý.

Có hai cách tiếp cận chung cho đối sánh ngữ nghĩa của dịch vụ là rõ ràng (explicit) và ngầm định (implicit). Tưởng tượng rằng các dịch vụ chứng khoán dùng các kí hiệu của thị trường chứng khoán New York đại diện cho tên công ty. Ví dụ "IBM" thì sẽ trả lại các thông tin và lược đồ quá trình làm ăn của công ty IBM trong 12 tháng vừa qua. Giả sử rằng có một vài dịch vụ thu phí của người sử dụng, một vài dịch vụ khác thì người dùng được cung cấp miến phí. Trong biểu diễn ngầm định, được đề xuất cho OWL-S, mô tả mỗi dịch vụ theo các IOPE, với phí là một trong những effect trong quá trình thực hiện của dịch vụ. Biểu diễn rõ ràng sử dụng một bản thể để mô tả mỗi dịch vụ như là một biến thể hiện với các lớp được định nghĩa như FreeStockProfiler hoặc FreeBasedStockProfiler.

Người dùng có thể trình bày rõ ràng các biểu diễn của dịch vụ mà họ yêu cầu đầy đủ chi tiết để đối sánh các dịch vụ tiềm năng khác, truyền các biểu diễn này tới các tác tử khác (tác tử đối sánh hoặc tác tử trung gian), và chấp nhận các gợi ý của các dịch vụ thỏa mãn yêu cầu. Chẳng hạn một camera cần được đặt gần máy in nhất để in ảnh mầu với chất lượng đủ tốt trong khoảng thời gian xác định. Sau đó nó trình bày rõ ràng biểu diễn thích hợp của máy in, camera có thể sử dụng mô tả để tìm kiếm danh bạ hoặc gửi bức ảnh tới một tác tử khác để đưa đến máy in thích hợp khác.

Một máy đối sánh (matching engine) nên có các khả năng:

- Hỗ trợ đối sánh ngữ nghĩa mềm dẻo trên các bản thể được sẻ chia
- Cung cấp việc điều khiển trên đối sánh tới dịch vụ yêu cầu
- Khuyến khích các nhà quảng cáo và người yêu cầu cần trung thực với biểu diễn dịch vụ của họ
- Hiệu quả và giảm thiểu số fasle negative và false positive trong phép đối sánh

Thuật toán đối sánh ngữ nghĩa có thể dựa trên OWL, do đó cho phép các tác tử đối sánh nhận ra độ tương tự ngữ nghĩa giữa yêu cầu và quảng cáo mặc dù cú pháp có thể khác nhau của các dịch vụ. Điều này đạt được bằng cách so sánh các IOPE được đặc tả trong mô hình dịch vụ với đặc tả trong yêu cầu. Để cung cấp khả năng đối sánh ngữ nghĩa mềm dẻo, yêu cầu được đối sánh theo phân cấp gộp bởi bản thể cho trước mà nó đã bao gồm các khái niệm bản thể được đối sánh hơn là sự tương tự về cú pháp giữa yêu cầu và quảng cáo. Do đó, bản thể cung cấp ngữ cảnh mà ở đó yêu cầu và quảng cáo được thể hiện rõ rang. Chẳng hạn, đối sánh

những chiếc xe ô tô con (car) trong quảng cáo với các xe ô tô (vehicle) thì các xe ô tô con được tính gộp (subsumed) vào xe ô tô.

Một đối sánh (match) giữa một quảng cáo và một yêu cầu xảy ra khi tất cả các đầu ra của yêu cầu khớp với đầu ra của quảng cáo và tất cả các đầu vào của quảng cáo khớp với đầu vào của yêu cầu. Chẳng hạn khi dịch vụ có khả năng đáp ứng các nhu cầu của người yêu cầu và người yêu cầu cũng cung cấp tất cả thông tin đầu vào cho dịch vụ đối sánh hoạt động. Do đó, nếu thậm chí một yêu cầu đầu ra không khớp với đầu ra của quảng cáo thì coi như đối sánh đã thất bại. Dựa vào sự tương ứng về ngữ nghĩa của yêu cầu quảng cáo, thì những kiểu đối sánh được phân biệt như sau:

- Đối sánh chính xác (exact): khi đầu ra của yêu cầu giống hệt với đầu ra của quảng cáo. Nghĩa là, đầu ra của quảng cáo thỏa mãn hoàn toàn với đầu ra của yêu cầu.
- Đối sánh Plug-in: khi đầu ra của yêu cầu được tính gộp (subsumed) với đầu ra của quảng cáo. Chẳng hạn, một dịch vụ cung cấp tất cả các loại xe ô tô là một plug in khớp (matched) với một yêu cầu xe ô tô con. Dịch vụ quảng cáo có thể được thay thế vị trí của dịch vụ yêu cầu. Việc đối sánh như vậy có thể làm giảm độ chính xác (precision) nhưng có khả năng thỏa mãn yêu cầu.
- Đối sánh tính gộp (subsume): Khi đầu ra của yêu cầu tính gộp cả đầu ra quảng cáo. Chẳng hạn một dịch vụ yêu cầu xe ô tô con khớp với phép tính gộp (subsuming) đối với một yêu cầu mong muốn xe ô tô. Việc đối sánh này sẽ làm giảm độ bao phủ (recall) bởi vì dịch vụ quảng cáo không hoàn toàn thỏa mãn yêu cầu.

Như vậy, đối sánh chính xác là trường hợp đặc biệt của đối sánh plug-in và đối sánh tính gộp.

9.3 LỰA CHỌN DỰA TRÊN MÔ HÌNH XÃ HỘI

Khám phá dịch vụ và xác định vị trí dịch vụ liên quan đến việc tìm cho ra một dịch vụ cho trước ở đâu. Đặc tả cho dịch vụ mong muốn tương ứng với một câu truy vấn. Tuy nhiên, thay vì câu trả lời đúng (correctness) thì ta xem xét câu trả lời đủ (completeness). Vài dịch vụ có thể thỏa mãn yêu cầu cụ thể trong khi tìm cho ra dịch vụ tốt nhất thì có thể rất khó khăn. Đây cũng là điểm mấu chốt phân biệt giữa rút trích thông tin và khám phá dịch vụ. Rút trích thông tin hầu như gặp khó khăn trong môi trưởng mở vì nguồn thông tin cần rút trích thường phải cụ thế (đóng). Tuy nhiên, khám phá dịch vụ không gặp khó khăn này vì trong kiến trúc hướng dịch vụ, bài toán khám phá và lựa chọn dịch vụ giảm thiểu nguồn thông tin cần khám phá và lựa chọn thành các dịnh vụ mong muốn (desired services).

Chắc chắn rằng việc tìm các dịch vụ mong muốn như vậy bao gồm các thao tác tìm hướng (navigation) thông tin (information navigation) trong nguồn thông tin được tổ chức trong một đồ thị. Một dịch vụ được khám phá bằng cách tìm kiếm trên đồ thị này. Theo truyền thống, những đồ thì này được xây dựng bởi các cạnh kết nối nguồn này đến nguồn khác. Ví dụ, các trang web chứa các siêu liên kết đến các trang web khác là khái niệm cơ bản nhất cho việc tìm hướng này. Tuy nhiên, web không cung cấp ngữ nghĩa hoặc khái niệm hợp lý. Do đó tất cả các cạnh ra của một nguồn thông tin thì được xem như nhau. Hiển nhiên rằng việc tìm

hướng đi đúng rất quan trọng khi tìm kiếm trên một đố thị khổng lồ với hàng triệu nút cũng như hàng tỉ cạnh, vì nó sẽ làm giảm đáng kể thời gian tìm kiếm.

9.3.1 Các kỹ thuật tư vấn

Tư vấn (recommendation) là công việc dự đoán nhu cầu hoặc mối quan tâm của người dùng. Các hệ tư vấn (recommender systems) đã được sử dụng rộng rãi trong lĩnh vực lựa chọn sản phẩm. Lọc cộng tác dựa trên nội dung là cách tiếp cận tĩnh cho việc lựa chọn dựa các trang web. Nó bao gồm lọc các trang web hoặc tài liệu theo các từ xuất hiện trong trang web hoặc tài liệu này. Tuy nhiên, đó là bước lùi của các tiêu chuẩn dịnh vụ web hiện nay, vì nó bao gồm các biểu diễn hình thức có cấu trúc của các dịch vụ, và hỗ trợ cho khám phá dựa trên các biểu diễn này.

Một cách tiếp cận khác nữa là lọc thông tin xã hội (social information filtering). Trong đó, lọc cộng tác (collaborative filtering) được sử dụng rộng rãi trong các trang web thương mại điện tử như amazon.com. Trong lọc cộng tác, các đánh giá của người dùng đối với các sản phẩm khác nhau được lưu trữ tập trung. Các đánh giá thường được thu thập khi người dùng mua sản phẩm. Người dùng được gợi ý mua sản phẩm nào đó dựa trên các đánh giá của người dùng khác. Ví dụ, cả Alice và Bob đã mua các cuốn sánh A,B,C và khi Alice mua cuốn sánh D thì hệ lọc cộng tác tư vấn cho Bob nên mua cuốn sách D.

Người tiêu dùng được gợi ý được gọi là người dùng tích cực (active user). Ý tưởng là làm sao đoán được một người dùng tích cực đánh giá một sản phẩm dựa trên đánh giá của người khác hay dựa trên đánh giá sản phẩm và dịch vụ khác. Cách tiếp cận thông thường là sử dụng tương quan Pearson (Pearson correlation) để đánh trọng số. Một cách tương ứng là sử dụng véc tơ tương tự giữa những người dùng mà các người dùng được mô hình hóa trong không gian nhiều chiều.

a. Tư vấn dựa trên mô hình

Theo cách tiếp cận này, một mô hình cho các người dùng cần được xây dựng trước và sau đó sử dụng mô hình này để đoán về người dùng tích cực. Sau khi gom nhóm (clustering) người dùng, người dùng tích cực sẽ xuất hiện trong một nhóm nào đó. Sau đó một mô hình xác suất ví dụ mạng Baysian để biểu diễn mô hình.

b. Tư vấn dựa trên bộ nhớ

Cách tiếp cận dựa trên bộ nhớ xem xét đánh giá tất cả các người dùng trực tiếp thay vì xen vào bước xây dựng mô hình. Việc dự đoán cho đánh giá của người dùng tích cực là tổng các trọng số của đánh giá từ người dùng khác mà ở đó trọng số sẽ tương ứng với độ tương tự giữa người dùng tích cức và mỗi người dùng khác. Có hai bước để dự đoán. Một là tính giá trị trung bình theo các đánh giá thu được, tiếp theo là tính độ tương tự hoặc không tương tự giữa đánh giá của các người dùng khác nhau.

9.3.2 Lựa chọn dịch vụ dựa trên tư vấn

Lựa chọn dịch vụ dựa trên tư vấn có thể được chia làm 3 khía cạnh:

- Khám phá bằng tìm kiếm (lookup)và tìm hướng (navigation) trong mạng.

- Tư vấn thông quá nhu cầu tương tự hoặc không tương tự
- Đánh giá chất lượng

Ba khía cạnh này có thể được xem xét trong một nền tảng duy nhất. Nền tảng đó là cộng đồng dịch vụ (service community). Một cộng đồng như vậy bao gồm một số người quan trọng (principals) mà mỗi người này có tiềm năng sử dụng và cung cấp các dịch vụ khác nhau. Những người quan trọng này được trợ giúp bởi các tác tử để giúp họ quản lý các tương tác. Các tác tử đánh giá dịch vụ và các lời tư vấn được cung cấp bởi người khác. Quan trọng hơn, giống như con người, các tác tử duy trì một danh bạ những người quan trọng đáng để tương tác, và quyết định ai có thể sử dụng dịch vụ. Vì vậy mỗi tác tử hỗ trợ chức năng đăng kí. Và cũng như vậy, các tác tử cũng có thể tương tác với nhau.

Một tác tử có thể truy vấn đến một tác tử khác về một dịch vụ mà nó cần. Tác tử nhận được truy vấn có thể lờ đi, hoặc đáp lại rằng sẽ thực hiện dịch vụ, hay gửi yêu cầu truy vấn tới tác tử khác để yêu cấu thực hiện dịch vụ.

TÀI LIỆU THAM KHẢO

[1] Munidar P. Singh, Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agent.* Wiley&Sons, 2005

[2] Liyang Yu. *Introduction to the Semantics Web and Semantics Web Services*. Capma Publisher, 2011.

[3] Bài giảng về Protégé: http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/ (truy cập ngày 24/12/2014).

[4] Ví dụ về OWL-S: http://www.daml.org/services/owl-s/1.1/examples.htlm (truy cập ngày 24/12/2014)

[5] Dịch vụ webASP.NET: http://www.tutorialspoint.com/asp.net/asp.net_web_services.htm (truy cập ngày 25/12/2014)

[6] Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, Shishir Narain, Ramratan Vennam. *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach. IBM Redbooks*, 2015.

[7] Sam Newman. *Building Microservices: Designing Fine-Grained Systems 1st Edition.* Wiley&Sons, 2015