BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC HUẾ

**TRƯỜNG ĐẠI HỌC KHOA HỌC**

**HỒ NGUYỄN THÀNH NHÂN**

TÌM HIỂU VỀ KIẾN TRÚC HƯỚNG DỊCH VỤ TRONG LĨNH VỰC CÔNG NGHỆ PHẦN MỀM VÀ ỨNG DỤNG

CHUYÊN NGÀNH:

MÃ SỐ:

KHOA HỌC MÁY TÍNH

60 48 01 01

LUẬN VĂN THẠC SĨ KHOA HỌC   
CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC

PGS. TS. HOÀNG HỮU HẠNH

Huế, 2016

# MỞ ĐẦU

Với sự phát triển của internet và với xu thế hội nhập chung của toàn thế giới, các tổ chức, doanh nghiệp bắt tay, phối hợp hoạt động và chia sẻ tài nguyên với nhau để nâng cao hiệu quả hoạt động. Khi đó các sản phẩm sẽ có độ phức tạp lớn hơn, kéo theo các vấn đề liên quan như chi phí sản xuất, chi phí quản lý và bảo trì. Bên cạnh đó, ngành công nghệ phần mềm còn phải đối mặt với các khó khăn trong xu thế mới như vấn đề tái sử dụng và mở rộng các hệ thống sẵn có, vấn đề về sự không tương thích giữa các hệ thống khác nhau của nhiều tổ chức...

Để giải quyết vấn đề trên, nhiều giải pháp đã được nghiên cứu và ứng dụng, nhưng hầu hết đều không giải quyết các khó khăn một cách triệt để và kết quả đạt được cũng không như mong đợi. Một giải pháp đang được cộng đồng CNTT rất quan tâm, đó là “Kiến trúc hướng dịch vụ” (Service Oriented Architecture – SOA. Đó là lý do tôi chọn hướng nghiên cứu của luận văn là **“Tìm hiểu về kiến trúc hướng dịch vụ trong lĩnh vực công nghệ phần mềm và ứng dụng”.**

Mục đích chính của luận văn là tìm một giải pháp để hỗ trợ cho việc lập trình một ứng dụng theo kiến trúc hướng dịch vụ.

Trong luận văn này, tôi giới thiệu một phương pháp để hỗ trợ việc lập trình theo kiến trúc hướng dịch vụ đó là xây dựng một Services Bus Plug-in dựa trên công nghệ nền tảng Eclipse.

# Chương 1 Tổng quan về kiến trúc hướng dịch vụ

## Công nghệ Web Services

### Tổng quan về Web Services

Web service là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng Internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML.

Xuất bản

Gởi/nhận thông điệp

Tìm kiếm

**Đăng ký dịch vụ (UDDI)**

**Người sử dụng dịch vụ**

**Nhà cung cấp dịch vụ**

Mô tả dịch vụ (WSDL)

Hình 1.1. Cơ chế hoạt động của Web Services

### Kiến trúc của Web Services

Kiến trúc của Web Services bao gồm các tầng như sau:

Hình 1.2. Kiến trúc của Web Services

### Các thành phần của Web Services

#### XML

#### WSDL – Web Services Description Language

#### UDDI – Universal Description, Discovery, and Integration

#### SOAP – Simple Object Access Protocol

## Kiến trúc hướng dịch vụ

### Kiến trúc hướng dịch vụ (SOA) là gì?

Kiến trúc hướng dịch vụ - SOA (Service Oriented Architecture) là một cách tiếp cận hay một phương pháp luận để thiết kế và tích hợp các thành phần khác nhau, bao gồm các phần mềm và các chức năng riêng lẻ lại thành một hệ thống hoàn chỉnh. Kiến trúc SOA rất giống với cấu trúc của các phần mềm hướng đối tượng gồm nhiều module. Tuy nhiên khái niệm module trong SOA không đơn thuần là một gói phần mềm, hay một bộ thư viện nào đó. Thay vào đó, mỗi module trong một ứng dụng SOA là một dịch vụ được cung cấp rải rác ở nhiều nơi khác nhau và có thể truy cập thông qua môi trường mạng, và mỗi module đóng vai trò là một “dịch vụ có tính kết nối lõng lẻo”. Nói một cách ngắn gọn, một hệ thống SOA là một tập hợp nhiều dịch vụ được cung cấp trên mạng, được tích hợp lại với nhau để cùng cộng tác thực hiện các tác vụ nào đó theo yêu cầu của khác hàng.

Dịch vụ (Service) là yếu tố then chốt trong SOA. Có thể hiểu dịch vụ như là một loại module thực hiện một quy trình nghiệp vụ nào đó.

Find

Register

Bind,

Execute

Hình 1.8. Mô hình tổng quan của SOA

### Các tính chất của một hệ thống SOA

#### Kết nối lỏng (Loose coupling)

Bên sử dụng dịch vụ không cần biết mọi thông tin chi tiết của dịch vụ trước khi triệu gọi nó thì quan hệ giữa 2 bên càng có tính lỏng lẻo. Tính kết nối lỏng lẻo giúp gỡ bỏ những ràng buộc điều khiển giữa những hệ thống đầu cuối.

#### Tái sử dụng dịch vụ

Bởi vì các dịch vụ được cung cấp trên môi trường mạng và được đăng ký ở một nơi nhất định nên chúng dễ dàng được tìm thấy và sử dụng lại. Thực ra tái sử dụng dịch vụ lại dễ dàng hơn tái sử dụng thành tố hay lớp. Những dịch vụ được dùng chung bởi tất cả các ứng dụng của một hệ thống SOA gọi là những dịch vụ chia sẻ cơ sở hạ tầng.

#### Quản lý chính sách

Tập các chính sách là tập tất cả các qui tắc chung mà mọi thành phần trong hệ thống đều phải tuân thủ. Khi sử dụng các dịch vụ chia sẻ trên mạng, tùy theo mỗi ứng dụng sẽ có một luật kết hợp riêng gọi là các chính sách.

#### Tự động dò tìm và ràng buộc động

SOA hỗ trợ khái niệm khai thác dịch vụ (service discovery). Một người sử dụng cần đến một dịch vụ nào đó có thể tìm kiếm dịch vụ dựa trên một số tiêu chuẩn khi cần. Người sử dụng chỉ cần hỏi một registry về một dịch vụ nào thỏa yêu cầu tìm kiếm.

#### Khả năng tự phục hồi

Một hệ thống tự phục hồi là hệ thống có khả năng tự phục hồi sau khi lỗi mà không cần sự can thiệp của con người.

#### Khả năng cộng tác

Kiến trúc hướng dịch vụ nhấn mạnh đến khả năng cộng tác (*Interoperability*), khả năng mà các hệ thống có thể giao tiếp với nhau trên nhiều nền tảng và ngôn ngữ khác nhau. Mỗi dịch vụ cung cấp một interface có thể được triệu gọi thông qua một dạng kết nối.

### Kiến trúc phân tầng chi tiết của SOA

Nhấn mạnh rằng, SOA là một phương pháp luận giúp chúng ta tận dụng sức mạnh của các nguồn lực, nguồn tài nguyên khác nhau trong mạng máy tính để trở thành một hệ thống nhất. Nên hiện tại không có mô hình thống nhất cho các thành phần của hệ thống SOA, mỗi công ty, tổ chức khi phát triển một hệ thống SOA có thể đưa ra mô hình các thành phần của SOA khác nhau. Đây là mô hình các thành phần của hệ thống SOA theo quan điểm của công ty IBM [6] và đây cũng là một mô hình khá phổ biến cho kiến trúc của hệ thống SOA.

Hình 1.10. Kiến trúc phân tầng của SOA

Enterprise Service Bus (ESB): Cung cấp khả năng kết nối cần thiết cho những dịch vụ trong toàn bộ hệ thống.

Dịch vụ tương tác (interaction services) Đây là những thành phần giúp các ứng dụng và người dùng cuối giao tiếp với nhau, ở đây người dùng cuối (enduser) không chỉ là con người, mà cũng có thể là cảm biến, robot,...

Dịch vụ quy trình (Process services) chịu trách nhiệm cho logic thành phần.

Dịch vụ thông tin (Information services) chịu trách nhiệm về tính logic của dữ liệu, dịch vụ này cung cấp các chức năng tập hợp, thay thế và chuyển đổi nhiều nguồn dữ liệu khác nhau được thực hiện bởi nhiều cách thức khác nhau.

Dịch vụ đối tác (Partner services) có trách nhiệm thu thập thông tin về đối tác (ví dụ như các chính sách và hạn chế) và cung cấp tài liệu, giao thức, các chức năng quản lý đối tác cho những quy trình nghiệp vụ có yêu cầu tương tác với đối tác bên ngoài và nhà cung cấp.

Dịch vụ ứng dụng nghiệp vụ (Business application services) chịu trách nhiệm về logic nghiệp vụ cốt lõi.

Dịch vụ truy cập (Access services) có trách nhiệm kết nối các ứng dụng và chức năng vào kiến trúc hướng dịch vụ. Cung cấp các chức năng bắc cầu cho những ứng dụng cũ (legacy applications), kho dữ liệu chính, và ESB nhằm kết hợp dịch vụ có trong những ứng dụng hiện tại vào hệ thống SOA.

Dịch vụ đổi mới và tối ưu hóa nghiệp vụ (Business innovation and optimization services) có trách nhiệm cung cấp các công cụ và các cấu trúc siêu dữ liệu để đại diện cho thiết kế nghiệp vụ, bao gồm cả chính sách và mục tiêu nghiệp vụ.

Dịch vụ phát triển (Development services) cung cấp môi trường tích hợp cho thiết kế và tạo ra giải pháp.

Dịch vụ cơ sở hạ tầng (Infrastructure services) Những dịch vụ này có trách nhiệm lưu trữ các ứng dụng SOA, giúp cung cấp sử dụng hiệu quả các nguồn tài nguyên để tối ưu băng thông, sẵn sàng và hiệu năng.

## Ngôn ngữ thi hành quy trình nghiệp vụ - BPEL

### Giới thiệu

Web Service Business Process Execution Language (viết tắt là WS-BPEL hay được gọi là BPEL) là một ngôn ngữ thi hành quy trình nghiệp vụ dùng để hỗ trợ phát triển các ứng dụng phức tạp, lớn đòi hỏi phải tổng hợp nhiều web services khác nhau.



Hình 1.11. Quy trình tích hợp với các dịch vụ đối tác

### Các khái niệm cơ bản

#### Nguyên tắc hoạt động của một tiến trình BPEL

Cốt lõi BPEL là khái niệm peer-to-peer là sự tương tác giữa các Web Services được mô tả trong WSDL. Một tiến trình BPEL làm thế nào để phối hợp giữa các dịch vụ khác nhau và kết hợp các dịch vụ đó lại thành một tiến trình hoàn chỉnh.

#### Cấu trúc của một tiến trình

Một tiến trình BPEL là một mô tả dưới dạng tài liệu XML.

<bpel:process name=*"test"* >

<!-- Import the client WSDL -->

<bpel:import location=*"testArtifacts.wsdl"*

namespace=*"http://eclipse.org/bpel/sample"*

importType=*"http://schemas.xmlsoap.org/wsdl/"* />

<bpel:partnerLinks></bpel:partnerLinks>

<bpel:variables> </bpel:variables>

<bpel:sequence name=*"main"*></bpel:sequence>

</bpel:process>

Hình 1.12. Cấu trúc file BPEL

## Tiểu kết chương 1

Qua các kiến thức tổng quan ta đã thấy rõ ràng kiến trúc hướng dịch vụ (SOA) là một kiểu kiến trúc có khả năng tái sử dụng lại các tài nguyên sẵn có, khả năng mở rộng và liên kết tốt với các hệ thống mới để tạo nên một môi trường đồng nhất, nó bao gồm các dịch vụ nghiệp vụ độc lập, không đồng nhất được kết hợp với nhau trong quy trình nghiệp vụ linh hoạt mềm dẻo. Công nghệ Web service cùng với ngôn ngữ thi hành quy trình nghiệp vụ - BPEL đã hiện thực hóa kiến trúc hướng dịch vụ (SOA), cho phép kết hợp các dịch vụ đơn lẻ và các hệ thống ứng dụng thành một quy trình nghiệp vụ đầy đủ.

# Chương 2 Khung ứng dụng hỗ trợ lập trình SOA



## Nền tảng Eclipse

### Giới thiệu

Eclipse là một nền tảng phát triển phần mềm mã nguồn mở, bao gồm một IDE (Intergrated Development Environment) và một hệ thống plug-in có khả năng mở rộng được. Trong khi hầu hết người dùng sử dụng Eclipse như là môi trường phát triển tích hợp Java (IDE), một số khác lại có nhu cầu mở rộng Eclipse. Do đó, Eclipse xây dựng PDE (Plug-in Development Environment) dành cho người dùng muốn mở rộng Eclipse. PDE cho phép người xây dựng các công cụ tích hợp vào môi trường Eclipse và mọi thứ trong PDE đều là plug-in.

### Các thành phần và kiến trúc

Kiến trúc của Eclipse được xây dựng dựa trên hai thành phần chính: thành phần lõi (*core*) và các thành phần gắn thêm (*plug-in*).

Thành phần lõi (core): bao gồm các chức năng, dịch vụ mà các hệ phát triển ứng dụng phải có như chức năng cung cấp giao diện, trình soạn thảo văn bản, gỡ lỗi… cần cho mọi nền tảng lập trình (cần cho các plug-in).

Thành phần gắn thêm (plug-in): bao gồm nhiều thành phần dễ dàng tích hợp vào nhiều ứng dụng chạy trên nền Eclipse. Các chức năng của thành phần core tách biệt với các chức năng của phần giao diện.

*EclipseSDK* bao gồm 3 phần chính:

Platform

Java Development Toolkit (JDT)

Plug-in Development Environment (PDE)

Với *JDT*, *Eclipse* là một môi trường hỗ trợ phát triển Java. JDT cũng có thể được coi như là một *Plug-in* cho *Eclipse* như là một môi trường lập trình tích hợp (*Java IDE -Integrated Development Enviroment*). PDE hỗ trợ việc mở rộng *Eclipse*, tích hợp các *Plug-in* vào *Eclipse Platform*. *Eclipse Platform* là nền tảng của toàn bộ phần mềm *Eclipse*, cung cấp những dịch vụ cần thiết cho việc tích hợp những bộ công cụ phát triển phần mềm dưới dạng *Plug-in*. *Eclipse Platform* được xây dựng dựa trên cơ chế phát hiện, tích hợp và chạy các thành phần gắn thêm. *Plug-in* là thành phần gắn thêm có thể hoạt động độc lập hoặc cùng với các thành phần khác. Thông thường, một ứng dụng có thể tổ hợp nhiều thành phần gắn thêm, mỗi thành phần gắn thêm này lại nối với các thành phần gắn thêm khác tùy vào mục đích sử dụng.



Hình 2.1. Kiến trúc tổng quan Eclipse

## Kiến trúc mô hình plug-in Eclipse

Plug-in trong Eclipse là một thành phần cung cấp một loại dịch vụ nhất định trong môi trường lập trình Eclipse [4]. Tương tự, Eclipse cung cấp cơ sở để hỗ trợ kết nối và thực thi các plug-in, hình thành một môi trường lập trình đồng nhất. Khi một chương trình Eclipse được thực thi, plug-in sẽ là một đối tượng trong class của plug-in. Class này sẽ cung cấp cấu hình và hỗ trợ quản lý cho các đối tượng plug-in. Class của plug-in phải được kế thừa từ org.eclipse.core.runtime.Plugin.

Trong cấu trúc thư mục của Eclipse, các plug-in đã được cài đặt sẽ được chứa trong thư mục plugins. Plug-in được mô tả bởi một tập tin manifest XML, gọi là plugin.xml. Tập tin manifest sẽ chỉ ra cách để Eclipse kích hoạt plug-in đó.

Hình 2.2. Minh họa một tập tin plug-in manifest

### Cài đặt và kích hoạt Plug-in

Để sử dụng một plug-in cần sao chép các tài nguyên tạo nên plug-in (tập tin manifest, tập tin jar, và các tập tin tài nguyên khác) vào trong thư mục plugins ở đường dẫn cài đặt Eclipse. Plug-in này sẽ được kích hoạt khi Eclipse thực thi chương trình và khi có yêu cầu thực hiện chức năng liên quan đến plug-in đó. Kích hoạt plug-in đồng nghĩa với nạp các class và khởi tạo các biến đối tượng ban đầu.

Trong mô hình Eclipse, một plug-in có thể liên quan đến plug-in khác theo dạng quan hệ sau:

Phụ thuộc (Dependency): Vai trò trong mối quan hệ này là plug-in phụ thuộc và plug-in tiên quyết. Một plug-in tiên quyết sẽ hỗ trợ chức năng cho plug-in phụ thuộc.

Mở rộng (Extension): Vai trò trong mối quan hệ này là plug-in chính và plug-in mở rộng. Plug-in mở rộng sẽ bổ sung chức năng cho plug-in chính.

Mối quan hệ này được đặc tả trong tập tin manifest thông qua các thành phần XML là *requires* và *extension*.

### Phụ thuộc – Dependency

Khi một plug-in phụ thuộc vào chức năng của các plug-in khác, mối quan hệ này sẽ được đặc tả bởi thuộc tính *requires* trong tập tin manifest của plug-in.

### Mở rộng – Extension

Khi các thành phần của plug-in được sử dụng bởi người sử dụng, một hay nhiều các giao diện người sử dụng (UI) của plug-in sẽ được thêm vào trong Eclipse. Quá trình này gọi là mở rộng (extension). Tuy nhiên, quá trình này không chỉ giới hạn cho các thành phần UI, mà còn cho cả các thành phần xử lý. Một extension được xác định bởi plug-in mở rộng (plug-in extender) và khiến plug-in chính (host plug-in) thay đổi chức năng của nó. Thông thường, việc thay đổi chức năng này có thể là thêm vào chức năng mới hoặc tùy chỉnh chức năng có sẵn thông qua các dịch vụ cung cấp bởi plug-in mở rộng.

#### Các thành phần của Extension

##### Plug-in chính (Host Plug-in)

Đối với một extension cụ thể, plug-in đóng vai trò chính sẽ cung cấp các điểm mở rộng (extension point) và được mở rộng. Bên cạnh việc cung cấp dịch vụ của riêng nó, plug-in này cũng làm nhiệm vụ điều phối và điều khiển các mở rộng khác.

##### Plug-in mở rộng (Extender plug-in)

Một plug-in có vai trò là mở rộng, khi nó thêm vào một số tính năng cho plug-in chính. Một mở rộng được khai báo bằng cách sử dụng thành phần XML *extension* trong file manifest.

##### Extension callback

Một đối tượng callback là một đối tượng thuần Java (không phải là plug-in) được gọi bởi plug-in chính trong một số trường hợp nhất định tương ứng với các extension-point được xác định. Giao diện của các đối tượng callback được cung cấp bởi plug-in chính. Để sử dụng các đối tượng callback này, cần khai báo class đối với một extension cụ thể. Vì việc sử dụng các đối tượng callback này là mở rộng tham chiếu đến giao diện callback, mà thông thường được đóng gói trong plug-in chính, nên có thể nói các plug-in mở rộng cũng phụ thuộc vào plug-in chính.

#### Mối quan hệ giữa Plug-in và Extension

Một hoạt động mở rộng là một khái niệm chung trong Eclipse và để hiểu rõ một cách khái quát, ta có thể tóm tắt mối quan hệ có thể tồn tại giữa đối tượng plug-in, extension-points và đối tượng callback.

Có thể có nhiều extension-point trong một plug-in chính.

Một plug-in có thể vừa là một plug-in chính, chỉ ra các extension-point, vừa là một plug-in mở rộng, mở rộng các plug-in khác.

Nhiều plug-in có thể mở rộng một extension-point cho trước.

Một plug-in có thể có nhiều extension-point.

Một plug-in mở rộng có thể bao gồm nhiều mở rộng của các plug-in chính khác nhau.

Một hành động mở rộng của một điểm mở rộng bởi một plug-in cụ thể có thể tạo ra nhiều đối tượng callback.

Một plug-in có thể xác định mở rộng thông qua extension-point của riêng nó.

#### Lược đồ extension-points

Một mở rộng cụ thể được xác định bởi một thành phần cấu hình XML trong plug-in mở rộng. Các thành phần này cung cấp các thông tin cần thiết để khởi tạo các đối tượng callback cần thiết cho phần mở rộng, cũng như các thông tin cần thiết để tùy chỉnh giao diện và hành vi của các plug-in chính. Khi một điểm mở rộng được tạo ra, ngoài việc khai báo trong file manifest của nó, cũng cần xác định cú pháp cấu hình cho phần mở rộng đó. Cú pháp này là một lược đồ XML và lưu trữ trong một tập tin có đuôi *.exsd.*

## Tiểu kết chương 2

Kiến trúc nền tảng Eclipse bao gồm nhiều thành phần. Các chức năng của thành phần lõi tách biệt với các chức năng của phần giao diện. Eclipse cung cấp các công cụ cùng cơ chế sử dụng và quy định để có thể tích hợp các công cụ một cách dễ dàng. Những cơ chế này được thể hiện thông qua giao diện API, lớp và phương thức. Ngoài ra, nền tảng Eclipse còn cung cấp các framework để dễ dàng phát triển các công cụ mới.

Eclipse SDK (Software development kit) gồm Eclipse Platform, JDT và PDE. JDT và PDE là các plug-in được gắn vào Platform. Eclipse Platform được xây dựng dựa trên cơ chế “*phát hiện, tích hợp và chạy*”. Plug-in là đơn vị chức năng nhỏ nhất có thể được phát triển và phân phối một cách riêng biệt. Thông thường, công cụ nhỏ được viết trong một plug-in riêng biệt và công cụ phức tạp được tách ra thành nhiều plug-in.

# Chương 3 Xây dựng ứng dụng trên nền tảng Eclipse



## Bài toán điều phối các lời gọi dịch vụ trong kiến trúc SOA

### Mục tiêu

Giải pháp này tập trung vào mở rộng nền tảng Eclipse cho phép phát triển và triển khai các Web Services như các plug-in khác trong Eclipse. Do đó, các dịch vụ được tích hợp trong một kịch bản định hướng đường ống để giải quyết các kịch bản nghiệp vụ cụ thể - dẫn tới việc ta xây dựng một kiến trúc hướng dịch vụ theo đường ống – Service Oriented Pipeline Architecture (SOPA).

### Giải pháp

Giải pháp là xây dựng một dịch vụ web đăng ký (web services registry), gọi là bus dịch vụ (Services Bus), phát triển như một plug-in Eclipse. Services Bus cung cấp các điểm mở rộng cho các nhà phát triển để xuất bản các lớp Java tiêu chuẩn của họ như các web services.

Một phần nữa trong giải pháp là triển khai một plug-in đường ống (Pipeline plug-in), đóng vai trò trung tâm trong việc điều phối dịch vụ của hệ thống và trong việc tạo ra các dịch vụ nghiệp vụ mới. Pipeline được định nghĩa bằng một cấu trúc XML để quy định các bước trong Pipeline và các chuyển đổi liên quan. Nó bao gồm một số câu lệnh điều kiện kiểu XSLT và trao đổi dữ liệu bởi biểu thức XPATH. Pipeline được thống nhất truy cập qua Web Service plug-in, quản lý toàn bộ hệ thống dịch vụ bao gồm plug-in services, pipelines và các dịch vụ web bên ngoài (external web services).

## Mô tả chi tiết

### Kiến trúc hướng dịch vụ theo đường ống

Chính thức thì SOPA={S,P}, nơi mà các dịch vụ S có thể là các giao diện người sử dụng dịch vụ (GUI services) Sv , web-services nội bộ Sw , và web-services từ bên ngoài Sx. Tức là S ={ Sv , Sw , Sx}. Pipelines P sẽ điều phối các dịch vụ nghiệp vụ khác nhau (Sw và Sx) và áp dụng các phép chuyển đổi T để trả lại kết quả cho người sử dụng hoặc dịch vụ khác, tức là P ={Sw , Sx ,T}.

### Services Bus

Phần cốt lõi của khung ứng dụng Eclipse là plug-in và cơ chế mở rộng của nó. Một plug-in có thể coi là điểm mở rộng để các plug-in khác có thể kết nối đến. Eclipse cũng cung cấp cơ chế để quản lý tập hợp các điểm mở rộng cơ bản và các đăng ký tương ứng của nó. Services Bus cung cấp các điểm mở rộng cho các nhà phát triển để xuất các lớp Java tiêu chuẩn của họ như các dịch vụ web. Cơ chế điểm mở rộng của Eclipse tạo điều kiện thuận lợi cho việc cấu hình các phần mở rộng với nhà cung cấp phần mở rộng. Khi khởi chạy ứng dụng, services bus sẽ tải tất cả các dịch vụ đã được kết nối với nhau và tự động triển khai chúng bằng cách sử dụng máy chủ nhúng Jetty và Apache AXIS. Các script triển khai được tạo ra từ các mô tả dịch vụ. Ngoài ra, Services Bus cũng sử dụng tiêu chuẩn WSDD và WSDL để cấu hình dịch vụ.

### Plug-n-Play Web Services

Điểm cơ bản trong phát triển Service Bus chính là đạt được mục tiêu của plug-n-play Web Services sử dụng các plug-in và cơ chế mở rộng của nền tảng Eclipse. Đầu tiên, một điểm mở rộng phải được cấu hình theo các đặc tả dịch vụ và các tiêu chuẩn triển khai như WSDL và WSDD.

### Tính trong suốt của lời gọi dịch vụ

Service Bus có thể được xem như cánh cửa để đi vào hệ thống SOPA dựa trên Eclipse. Nó có chức năng định tuyến các yêu cầu dịch vụ đến các dịch vụ được kết nối hiện thời. Một chức năng khác quan trọng của Service Bus đó là nâng cao tính linh hoạt của hệ thống SOPA. Từ đó, nó cung cấp một lớp truy cập đồng nhất và trong suốt đến các dịch vụ bên trong và dịch vụ bên ngoài hệ thống.

Lời gọi dịch vụ (Service call): Các dịch vụ cắm trong Service Bus có thể được gọi bởi các class tương ứng hoặc sử dụng máy chủ Apache AXIS.

### Dịch vụ đường ống – Services Pipeline

Một pipeline trong thuật ngữ SOPA là một tập hợp có tên duy nhất các lời gọi service và chuyển đổi trung gian. Pipeline plug-in cho phép hệ thống SOPA hiểu rõ kịch bản dựa trên dịch vụ cơ bản và các pipelines. Ý tưởng dựa trên các thành phần đường ống (component pipeline), mỗi thành phần trong pipelines chỉ định một hoạt động cụ thể, các thành phần móc nối với nhau vào pipelines mà không cần yêu cầu lập trình.

Các dịch vụ có sẵn trong môi trường SOPA được định tuyến thông qua Services Bus plug-in, tức là tất cả dịch vụ sẽ được yêu cầu từ Services Bus, nơi chịu trách nhiệm cho việc tìm thấy và triệu gọi các dịch vụ tương ứng để làm nhiệm vụ. Tính năng này cung cấp sự trong suốt của dịch vụ trong toàn bộ môi trường SOPA.

Plug-in Services

Services

Services

Pipelines



External

Web Services

Services Bus

Plug-in



Tính trong suốt của dịch vụ trong SOPA

### Tính năng kỹ thuật và các loại kịch bản của Pipeline

#### Đa lời gọi

Một pipeline có thể có nhiều lời gọi đến các dịch vụ SOPA. Kết quả trả về có thể kết hợp với nhau để tạo ra tập các kết quả cho từng loại dịch vụ.

#### Lời gọi lồng nhau

Cũng có thể sử dụng những lời gọi lồng trong SOPA.

#### Lời gọi có điều kiện

Pipeline plug-in cung cấp các loại thẻ cho phép gọi đến các dịch vụ dựa trên điều kiện kiểu XPATH. Để thực hiện điều này, có hai cấu trúc điều kiện được sử dụng là “if” và “choose”. Ở cả hai cấu trúc này, điều kiện dạng XPATH sẽ được kiểm tra và dựa trên kết quả của dịch vụ tương ứng được gọi.

#### Các tham số Xpath trích xuất

Hiện tại chúng ta đã giới thiệu ba loại tham số cho các lời gọi dịch vụ:

Literal (dạng trực tiếp): truyền các giá trị tham số trực tiếp vào dịch vụ.

Pipeline parameter: các giá trị tham số được thay bằng các giá trị tham số pipeline trong quá trình chạy.

Service call parameter: kết quả của dịch vụ này là đầu vào cho dịch vụ khác.

#### Pipes trong pipes

Một ưu điểm của việc kết hợp các dịch vụ thành pipeline là việc tạo ra dịch vụ phức dựa trên các dịch vụ sẵn có. Các pipeline này có thể được gọi và tái sử dụng bởi các pipeline khác hoặc các thành phần SOPA thông qua tên gọi.

## Tiểu kết chương 3

Việc xây dựng kiến trúc hướng dịch vụ theo đường ống – SOPA đã giúp các dịch vụ có thể giao tiếp với nhau, tích hợp lại thành các quy trình nghiệp vụ mới đáp ứng yêu cầu của người sử dụng. Đặc biệt là nó đã hỗ trợ cho các lập trình viên thuận tiện hơn trong việc sử dụng và khai báo các tập dịch vụ có sẵn để hình thành nên các quy trình nghiệp vụ mới phục vụ cho lợi ích của các doanh nghiệp. Một vấn đề phát triển và nghiên cứu quan trọng trong việc xây dựng plug-in đó là làm đơn giản quá trình tạo pipeline, dựa trên các yêu cầu đến và các dịch vụ hiện có để hình thành nên dịch vụ mới.

# KẾT LUẬN

Hướng tiếp cận SOA là một trong những lựa chọn tối ưu trong công nghệ phần mềm hiện nay, đặc biệt cho các doanh nghiệp công nghệ phần mềm hiện nay trong việc xây dựng lại các kiến trúc hệ thống có tính linh hoạt, mềm dẻo, tái sử dụng cao. Để triển khai kiến trúc hướng dịch vụ và công nghệ Web Services việc lựa chọn nền tảng Eclipse đáp ứng được tính mềm dẻo và linh hoạt của nó. Luận văn đã tìm hiểu chi tiết kiến trúc Eclipse hỗ trợ kiến trúc hướng dịch vụ trên cơ sở đó xây dựng nền tảng cho kiến trúc plug-in, làm gia tăng tính mềm dẻo trong các kiến trúc phần mềm. Bên cạnh đó, luận văn đã trình bày những khái niệm tổng quan về công nghệ Web Services, những ưu điểm của công nghệ này trong việc giải quyết bài toán tích hợp giữa các hệ thống, cũng như khả năng ứng dụng của nó trong việc xây dựng mô hình kiến trúc hướng dịch vụ. Luận văn đã xây dựng được một hệ thống điều phối dịch vụ dựa trên kiến trúc plug-in trên nền tảng công nghệ của Eclipse để có thể triển khai lập trình ứng dụng theo kiến trúc hướng dịch vụ.

Hướng nghiên cứu tiếp theo của luận văn là tiếp tục hoàn thiện hơn nữa hệ thống plug-in; thử nghiệm với nhiều Web Services phức tạp hơn, đồng thời nghiên cứu tăng cường khả năng bảo mật.

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. Nguyễn Trọng Dũng, *Giáo trình Java – Chương 3:* *Phát triển Web Services với các công nghệ chuẩn của java* , Trường ĐHSP Hà Nội.
2. Hồ Bảo Thanh, Nguyễn Hoàng Long (2005), *Nghiên cứu kiến trúc hướng dịch vụ (Service-Oriented Architecture) và ứng dụng,* Luận văn cử nhân tin học khoa CNTT – ĐH KHTN Tp.HCM.

**Tiếng Anh**

1. Abdaldhem Albreshne, Patrik Fuhrer, Jacque Pasquier Spasquier (2009), *Web Services Technologie: State of the Art,* Definitions, Standards, Case Study, pp. 1-22.
2. Azad Bolour (2003), “Notes on the Eclipse Plug-in Architecture”*,* Bolour Computing, <http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html>, (Ngày truy cập 4/4/2016).
3. Dr Alex Blewitt (2013), *Eclipse 4 Plug-in Development by Example*, Packt Publishing, pp. 7-34.
4. [Bertrand Portier](http://www.ibm.com/developerworks/webservices/library/ws-soa-term1/index.html#authorN1001E) (2007), “SOA terminology overview”, IT Architect IBM, <http://www.ibm.com/developerworks/library/ws-soa-term1/> (truy cập ngày 4/4/2016).
5. David Gallardo (2002), “Developing Eclipse plug-ins”, IBM <http://www.ibm.com/developerworks/opensource/library/os-ecplug/> (truy cập ngày 4/4/2016).
6. Eric Clayberg, Dan Rubel (2009), *Eclipse Plug-ins*, Third Edition, pp. 21-153, 657-680.
7. Hartwig Gunzer (2002), *Introduction to Web Services*, Sales Engineer, Borland.
8. Kiet T. Tran, bookcoon.com (2013), *Introduction to Web Services with Java*, 1st edition, pp. 15-115.
9. Mark D. Hansen (2007), *SOA Using Java Web Services*, Printed in the United States of America.
10. Martin Kalin (2013), *Java Web Services : Up and Running*, Second Edition, O’Reilly Media, pp. 1-9, 291-329.
11. Yuli Vasiliev (2007), *SOA and WS-BPEL*, Packt publishing, pp. 5-37.
12. Chris Aniszczyk, David Gallardo, “Get Started with the Eclipse platform”, <http://www.ibm.com/developerworks/opensource/library/os-eclipse-platform/#N101FF>
13. OASIS (2007), “Web Services Business Process Execution Language Version 2.0”, <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>.