Kiến trúc hướng dịch vụ (webservice)

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Giới thiệu

Trong khi các nhà quản lý công nghệ thông tin đang phải đối đầu với việc giảm giá thành và tối đa lợi ích của các công nghệ hiện có, họ vẫn phải liên tục cố gắng để phục vụ khách hàng được tốt hơn, trở nên cạnh tranh hơn và phản ứng nhanh hơn với chiến lược cửa doanh nghiệp.

Có hai yếu tố chính ẩn sau những áp lực này, đó là tính không đồng nhất của các hệ thống, ứng dụng và kiến trúc khác nhau với thời gian tồn tại và công nghệ khác nhau. Việc tích hợp nhiều sản phẩm từ nhiều nhà cung cấp và nhiều nền tảng thực sự là điều khó khăn. nhưng chúng ta cũng không thể cố gắng tiếp cận theo kiểu một nhà cung cấp công nghệ thông tin vì các bộ ứng dụng và kiến trúc hỗ trợ rất không mềm dẻo.

Sự thay đổi về mặt công nghệ là yếu tố thứ hai mà các nhà quản lý công nghệ thông tin phải đối mặt. Sự toàn cầu hoá và kinh doanh điện tử đang làm tăng tốc sự thay đổi và dẫn đến sự cạnh tranh gay gắt trong việc rút ngắn chu kỳ sản xuất để có thể chiếm ưu thế đối với đối thủ cạnh tranh. Các thay đổi về yêu cầu và nhu cầu của khách hàng nhanh chóng hơn do tác động của phân phối cạnh tranh và sự phong phú về thông tin sản phẩm và dịch vụ diễn ra nhanh hơn.

Các cải tiến trong công nghệ tiếp tục tăng nhanh, làm tăng sự thay đổi yêu cầu của khách hàng. Doanh nghiệp phải nhanh chóng thích nghi để tồn tại, chưa kể đến việc phải thành công trong môi trường cạnh tranh đông ngày nay, và hạ tầng công nghệ thông tin phải đem lại khả năng thích nghi cho các doanh nghiệp.

Vì vậy, các tổ chức kinh doanh đang phát triển từ sự phân chia doanh nghiệp theo chiều dọc, cô lập của những năm 1980 về trước thành các cấu trúc chú trọng quy trình kinh doanh theo chiều ngang của những năm 1980, 1990, tới mô hình kinh doanh mới trong đó các doanh nghiệp có tác động lẫn nhau. Các dịch vụ doanh nghiệp hiện nay cần phải được thành phần hoá và phân tán. Cần trú trọng vào dây chuyền cung cấp mở rộng, cho phép khách hàng và đối tác truy cập tới các dịch vụ kinh doanh.

Làm thế nào để môi trường công nghệ thông tin trở nên linh hoạt và nhạy bén hơn đối với sự thay đổi của các yêu cầu kinh doanh? Làm thế nào để các hệ thống và ứng dụng không đồng nhất trao đổi với nhau một cách liền mạch? Làm thế nào để đạt mục tiêu kinh doanh mà không bị phá sản các doanh nghiệp?

Đã có nhiều giải pháp được đề ra song song với sự phát triển của các vấn đề kinh doanh: kiến trúc các đơn vị riêng lẻ,có cấu trúc, mô hình khách chủ, mô hình ba lớp, mô hình đa lớp, mô hình đối tượng phân tán, các thành phần, dịch vụ. Hiện nay, nhiều nhà quản lý và các chuyên gia công nghệ thông tin tin rằng chúng ta đang tiến ngày một gần hơn tới câu trả lời với kiến trúc hướng dịch vụ (SOA).

Để đáp ứng được các yêu cầu về sự không đồng nhất, tính liên thông và sự thay đổi yêu cầu không ngừng, một kiến trúc như vậy phải đem lại một nền tảng cho việc xây dựng dịch vụ ứng dụng các đặc tính sau:

- Liên kết lỏng lẻo (Loosely Couple).

- Vị trí trong suốt (Location Transparent).

- Độc lập về giao thức (Protocol Independent)

Dựa trên một kiến trúc hướng dịch vụ như vậy, người dùng dịch vụ thậm chí không cần quan tâm tới một dịch vụ cụ thể mà mình đang trao đổi thông tin vì hạ tầng cơ sở, hay là tuyến dịch vụ (Services bus), sẽ tạo ra một lựa chọn thích hợp cho người dùng. Các đặc tả kỹ thuật cụ thể từ các công nghệ cài đặt khác nhau như J2EE hay .NET không làm ảnh hưởng tới người dùng SOA. Người dùng cũng có thể cân nhắc và thay thế một cài đặt dịch vụ tốt hơn nếu tồn tại một dịch vụ khác có chất lượng tốt hơn.

Định nghĩa về kiến trúc hướng dịch vụ (HDV)

Thuật ngữ kiến trúc hướng dịch vụ (SOA–Service Oriented Architectural) tuy không mới nhưng gần đây được nói đến rất nhiều, đặc biệt khi Microsoft cho ra đời công nghệ .NET, mà ở đó việc phát triển các dịch vụ (Service) trở nên dễ dàng hơn bao giờ hết.

Tùy theo quan điểm người dùng, hiện có rất nhiều định nghĩa khác nhau về kiến trúc hướng dịch vụ. Ví dụ về một định nghĩa của kiến trúc hướng dịch vụ:

"Kiến trúc hướng dịch vụ là một cách tiếp cận để tổ chức các tài nguyên công nghệ thông tin mà ở đó dữ liệu, logic và nguồn lực hạ tầng được truy cập thông qua các giao diện (interface) và thông điệp (Message)."

Xét về khía cạnh sử dụng thì có thể hiểu đơn giản như sau: *"Dịch vụ là một tập các chương trình con (thư viện) cho phép các chương trình chạy trên các máy tính khác trong mạng có thể triệu gọi và sử dụng..."*

Nói tóm lại, SOA có thể được hiểu là một hướng tiếp cận để xây dựng các hệ thống phân tán cung cấp các chức ứng dụng dưới các dạng dịch vụ tới các ứng dụng người cuối cùng hoặc các dịch vụ khác:

SOA là một kiến trúc dùng trong các chuẩn mở để biểu diễn các thành phần mềm như là các dịch vụ.

Cung cấp một cách thức chuẩn hoá cho việc biểu diễn và tương tác với các thành phần phần mềm.

Các thành phần phần mềm riêng lẻ trở thành các khối cơ bản để có thể sử dụng lại để xây dựng các ứng dụng khác.

Được sử dụng để tích hợp các ứng dụng bên trong và bên ngoài tổ chức.

Dịch vụ là yếu tố then chốt trong SOA. Có thể hiểu dịch vụ như là hàm chức năng (modul phần mềm) thực hiện theo quy trình nghiệp vụ nào đó. Các dịch vụ trong SOA có đặc điểm sau:

1. Các dịch vụ là có thể tìm kiếm được.
2. Các dịch vụ có tính liên thông.
3. Các dịch vụ không được gắn kết chặt chẽ với nhau.
4. Các dịch vụ là phức hợp, bao gồm nhiều thành phần, được đóng gói ở mức cao.
5. Các dịch vụ trong suốt về vị trí.
6. Các dịch vụ có khả năng tự hàn gắn.

Một cách cơ bản, SOA là tập hợp các dịch vụ kết nối " mềm dẻo " với nhau (nghĩa là một ứng dụng có khả năng giao tiếp với một ứng dụng khác mà không cần biết các chi tiết hệ thống bên trong), có giao diện được định nghĩa rõ ràng và độc lập với nền tảng của hệ thống, và có thể tái sử dụng. SOA là cấp độ cao hơn của sự phát triển ứng dụng, chú trọng đến quy trình nghiệp vụ và dùng giao diện chuẩn để che dấu sự phức tạp kỹ thuật bên dưới.

Thiết kế SOA tách riêng phần thực hiện dịch vụ (phần mềm) với giao diện gọi dịch vụ. Điều này tạo nên một giao diện nhất quán cho ứng dụng sử dụng dịch vụ mà không cần quan tâm tới công nghệ thực hiện dịch vụ. Thay vì xây dựng các ứng dụng đơn lẻ và đồ sộ, nhà phát triển sẽ xây dựng các dịch vụ tinh gọn hơn có thể triển khai và tái tạo sử dụng trong toàn bộ quy trình nghiệp vụ. Điều này cho phép tái sử dụng phần mềm tốt hơn, cũng như tăng sự mềm dẻo vì các nhà phát triển có thể cải tiến dịch vụ mà không làm ảnh hưởng đến ứng dụng sử dụng dịch vụ.

Triêt lý SOA không hoàn toàn mới, DCOM và CORBA cũng có kiến trúc tương tự. Tuy nhiên, các kiến trúc cũ ràng buộc các thành phần với nhau quá chặt ví dụ như các ứng dụng phân tán muốn làm việc với nhau phải được thoả thuận về chi tiết tập hàm API, một thay đổi mã lệnh trong thành phần COM sẽ yêu cầu những thay đổi tương ứng đối với mã lệnh truy cập thành phần DCOM này.

Ưu điểm lớn nhất của SOA là khả năng kết nối mềm dẻo và tái sử dụng. Các dịch vụ có thể được sử dụng trên nền tảng bất kỳ và được viết với ngôn ngữ bất kỳ (ví dụ, ứng dụng Java có thể liên kết với dịch vụ mạng. NET và ngược lại).

SOA dựa trên hai nguyên tắc thiết kế quan trọng :

- Modul: tách vấn đề lớn thành nhiều vấn đề nhỏ.

- Đóng gói: che dấu dữ liệu và logic trong từng modul đối với truy cập từ bên ngoài.

Một thiết kế kiến trúc phù hợp với khái niệm của SOA cần tuân theo những tính chất sau:

Một dịch vụ là một đơn vị phần mềm gồm các hoạt động nghiệp vụ có tính chứa đựng và mức độ đóng gói cao (coarse-grained).

Một dịch vụ có thể dùng lại được, cho phép có thể xây dựng được một dịch vụ mới từ các dịch vụ hiện có. Do đó, việc quan sát các hàm ý có thể có của các thuộc tính phi chức năng như tính giao dịch là rất quan trọng.

Một giao diện dịch vụ là một điểm cuối mạng (Network Endpoint) đảm bảo tính độc lập và trong suốt về vị trí.

Một dịch vụ cần có khả năng được phát hiện ra một cách công khai bằng cách sử dụng một nơi đăng ký dịch vụ nhằm cho phép các liên kết động tới dịch vụ.

Một dịch vụ cần đảm bảo tính liên thông bằng cách hỗ trợ các giao thức truyền thông được chuẩn hoá và các định dạng dữ liệu rõ ràng.

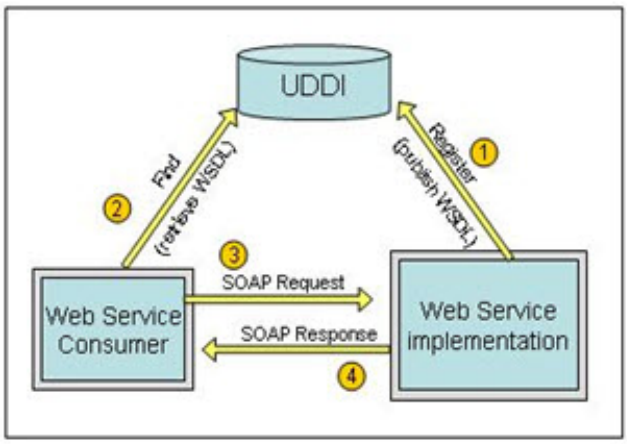
Các đặc điểm trên đảm bảo cho một kiến trúc hướng dịch vụ khả năng gắn kết lỏng lẻo của các dịch vụ phân tán và có tính modul bằng cách sử dụng các giao ước dịch vụ để mô tả các định dạng thông điệp cần thiết.

Các thành phần và mô hình trong kiến trúc HDV

Với mục tiêu là xây dựng một kiến trúc dịch vụ thực sự đơn giản và tính tương thích cao, SOA được xây dựng dựa trên các chuẩn rất phổ biến là SOAP (Simple Object Access Protocol – giao thức truy xuất đối tượng đơn giản) và XML (eXtensible Makup

Language – ngôn ngữ đánh dấu mở rộng). Hai chuẩn này đóng vai trò là thành phần xương sống để truyền nhận các thông điệp giữa các đối tượng truyền thông. Tức là, bất kỳ thành phần nào nếu hiểu được giao thức này thì hoàn toàn có thể sử dụng các dịch vụ mà không phụ thuộc vào ngôn ngữ lập trình hay hệ điều hành đang sử dụng.

Mô hình của kiến trúc có thể mô tả như sau:



Mô hình kiến trúc hướng dịch vụ

Các thực thể trong kiến trúc hướng dịch vụ bao gồm:

1. Dịch vụ (Services).
2. Thành phần sử dụng dịch vụ (Services consumer/ Client / Request).
3. Thành phần cung cấp dịch vụ (Services Provider).
4. Thành phần đăng ký dịch vụ (Services Registry).
5. Giao ước dịch vụ (Contract).
6. Uỷ nhiệm dịch vụ.
7. Ràng buộc sử dụng dụng dịch vụ (Services lease).

Dịch vụ:

Dịch vụ chứa một chức năng rõ ràng, tự chứa đựng và không phụ thuộc vào ngữ cảnh hay trạng thái của các dịch vụ khác. Các thành phần sử dụng dịch vụ có thể truy cập tới dịch vụ thông qua giao diện dịch vụ được xuất bản. Dịch vụ có các tính chất sau:

1. Dịch vụ có tính chất rõ ràng, là một đơn vị chức năng nghiệp vụ có thể được triệu gọi
2. Có khả năng triệu gọi thông qua các giao thức truyền thông chung.
3. Có tính liên thông và vị trí trong suốt.
4. Dịch vụ được định nghĩa bằng các giao diện tường minh.
5. Các giao diện độc lập với cài đặt.
6. Cung cấp giao ước giữa các thành phần cung cấp và sử dụng dịch vụ.

7. Dịch vụ là các modul phức tạp, bao gồm nhiều thành phần. Mức độ đóng gói của dịch vụ càng cao thì dịch vụu càng có khả năng tái sử dụng và linh hoạt.

Thành phần sử dụng dịch vụ(Web Services Customer):

Thành phần sử dụng dịch vụ là một ứng dụng, một dịch vụ, hoặc một loại modul phần mềm khác có yêu cầu sử dụng dịch vụ. Đây là thực thể khởi tạo việc định vị định vụ tại một kho đăng ký dịch vụ, liên kết tới dịch vụ qua một kênh truyền thông và thực thi các chức năng của dịch vụ. Thành phần náy thực thi nhiệm vụ bằng cách gửi tới dịch vụ một yêu cầu được định dạng theo đúng giao ước.

Thành phần cung cấp dịch vụ(Web Services Implementation):

Thành phần cung cấp dịch vụ là một thực thể có khả năng được địa chỉ hoá qua mạng, nó có thể chấp nhận và thực thi các yêu cầu từ những thành phần sử dụng dịch vụ. Thành phần cung cấp dịch vụ có thể là một hệ thôngs máy tính lớn, một thành phần, hoặc một loại hệ thống phần mềm khác có thể thực thi các yêu cầu dịch vụ. Thực thể này xuất bản giao ước dịch vụ của nó trong một kho đăng ký dịch vụ để các thành phần sử dụng dịch vụ có thể truy cập.

Thành phần đăng ký dịch vụ(UDDI):

Thành phần đăng ký dịch vụ là một thư mục trên mạng có chứa các dịch vụ sẵn dùng. Đây là một thực thể chấp nhận và lưu trữ các giao ước từ các thành phần cung cấp dịch vụ và cung cấp các giao ước đó cho các thành phần sử dụng dịch vụ.

Dịch vụ UDDI, đóng vai trò như người chỉ đường cho các thành phần sử dụng dịch vụ, tạo sự trong suốt về vị trí đối với thành phần cung cấp dịch vụ; để khi thành phần này thay đổi thì thành phần sử dụng dịch vụ không cần phải biên dịch hay cấu hình lại.

Giữa ba thành phần: cung cấp dịch vụ, sử dụng dịch vụ, đăng ký dịch vụ, việc trao đổi dữ liệu hoàn toàn dùng giao thức SOAP và nội dung bên trong được định dạng bằng XML.

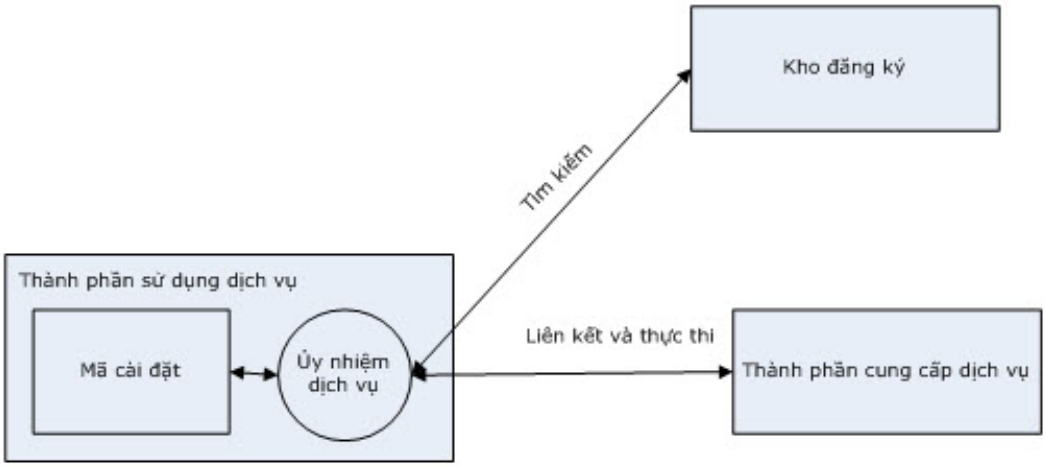
Giao ước dịch vụ:

Một giao ước là một bản đặc tả cách thức để thành phần sử dụng dịch vụ có thể tương tác với thành phần cung cấp dịch vụ. Nó chỉ ra khuôn dạng của thông điệp yêu càu và thôgn điệp đáp ứng từ các dịch vụ. Giao ước dịch vụ có thể đòi hỏi một tập các điều kiện tiên quyết và điều kiện sau. Các điều kiện này xác định trạng thái cần thiết của dịch vụ để thực thi một chức năng cụ thể. Bản giao ước này cũng có thể bao gồm các mức độ chất lượng của dịch vụ, các đặc tả cho các khoá chạnh phi chức năng của dịch vụ.

Uỷ nhiệm dịch vụ:

Thành phần cung cấp dịch vụ cung cấp một uỷ nhiệm dịch vụ cho thành phần sử dụng dịch vụ. Thành phần sử dụng dịch vụ thực thi các yêu cầu bằng cách gọi một hàm API trên nó. Uỷ nhiệm dịch vụ (hình 1.9) sẽ tìm một giao ước và một tham chiếu tới thành phần cung cấp dịch vụ trong nơi đăng ký. Sau đó, nó định dạng thông điệp yêu cầu và thực thi yêu cầu trên danh nghĩa của thành phần sử dụng dịch vụ. Uỷ nhiệm dịch vụ là một thực thể không bắt buộc, nó chỉ đơn giản hoá cho thành phần sử dụng dịch vụ và thành phần sử dụng dịch vụ hoàn toàn có thể viết phần mềm để truy cập tới dịch vụ.

Một thành phần cung cấp dịch vụ sẽ cung cấp nhiều uỷ nhiệm cho các môi trường khác nhau, mỗi uỷ nhiệm dịch vụ được viết bằng ngôn ngôn ngữ của các thành phần sử dụng dịch vụ. Ví dụ, một thành phần cung cấp dịch vụ có thể cung cấp các uỷ nhiệm dịch vụ cho Java, Visual Basic, Delphi nếu đó là các nền tảng của các thành phần sử dụng dịch vụ. Mặc dù uỷ nhiệm dịch vụ là không bắt buộc nhưng có thể cải thiện một cách đáng kể hiệu năng và tính tiện dụng cho các thành phần sử dụng dịch vụ.



Uỷ nhiệm dịch vụ

Ràng buộc sử dụng dịch vụ:

Ràng buộc sử dụng dịch vụ mà các thành phần đăng ký dịch vụ gán cho thành phần sử dụng dịch vụ rất cần thiết để dịch vụ bảo trì được thông tin trạng thái liên kết giữa thành phần sử dụng và thành phần cung cấp. Nó tạo ra sự gắn kết không chặt chẽ giữa các thành phần này bằng cách giới hạn khoảng thời gian mà chúng được liên kết với nhau. Không ràng buộc, một thành phần sử dụng dịch vụ có thể liên kết với một dịch vụ mãi mãi và không bao giờ liên kết lại với các giao ước của nó.

Các thao tác trong kiến trúc hướng dịch vụ bao gồm:

Xuất bản dịch vụ: để có thể truy cập được, mô tả dịch vụ phải được xuất bản để nó có thể được tìm thấy và triệu gọi bởi một người dùng dịch vụ.

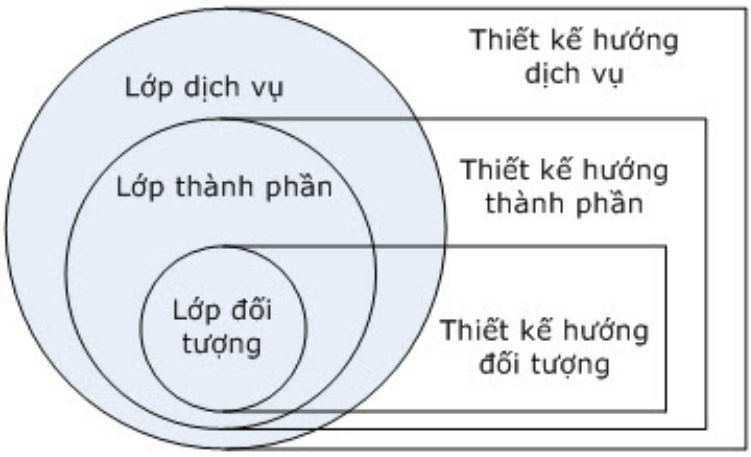
Tìm kiếm dịch vụ: một người yêu cầu dịch vụ định vị một dịch vụ bằng cách yêu cầu nơi đăng ký dịch vụ một dịch vụ phù hợp với các tiêu chí đặt ra.

Liên kết và thực thi dịch vụ: sau khi nhận được mô tả dịch vụ, người dùng sẽ gọi dịch vụ theo các thông tin mô tả.

Như hình 5.2, đầu tiên thành phần cung cấp dịch vụ sẽ đăng ký dịch vụ của mình với một UDDI service (thông tin đăng ký được đặc tả theo định dạng gọi là ngôn ngữ đặc tả dịch vụ – WSDL), khi đã đăng ký rồi thì thành phần sử dụng dịch vụ nếu muốn sử dụng thì sẽ truy xuất UDDI Service để lấy thông tin về vị trí service hiện đang đặt trên địa chỉ nào, sau đó thực hiện request service mong muốn và sau khi chạy service đó sẽ response kết quả về cho phía sử dụng dịch vụ.

Xây dựng ứng dụng theo kiến trúc HDV.

Trong sự tiến hoá của các kỹ thuật xây dựng phần mềm, kỹ thuật lập trình hướng đối tượng thích hợp để cài đặt các thành phần. Trong khi các thành phần lại là cách thích hợp nhất để cài đặt các dịch vụ, mặc dù cần hiểu rằng một ứng dụng dựa thành phần tốt không cần thiết là một ứng dụng dịch vụ tốt. Khi vai trò của dịch vụ trong kiến trúc ứng dụng được hiểu rõ, chúng ta có thể tích hợp các thành phần mới và các thành phần hiện có. Hình 1.14 minh hoạ cách cài đặt ở mức độ đóng gói cao hơn khi tiến gần tới người sử dụng, nó cũng cho thấy dịch vụ là cách thích hợp để thể hiện khung nhìn ngoài của hệ thống với sự tái sử dụng bên trong có sử dụng thiết kế thành phần truyền thống.



Các tầng cài đặt trong thiết kế: đối tượng, thành phần, dịch vụ

Các nguyên tắc trong thiết kế hướng dịch vụ

Việc tiếp cận xây dựng các hệ thống dựa trên mô hình hướng dịch vụ phải tuân theo bốn nguyên tắc sau:

Nguyên tắc 1: Giao diện của dịch vụ phải rõ ràng. Các dịch vụ tương tác qua việc truyền đi các thông điệp tường minh. Chúng ta không cần biết về không gian nằm sau giao diện của dịch vụ. Vượt qua các giao diện của dịch vụ có thể tốn nhiều công sức và chi phí. Các giao diện rõ ràng cho phép cài đặt các tương tác độc lập- nghĩa là không cần biết về nền tảng hay ngôn ngữ lập trình được lựa chọn để cài đặt.

Nguyên tắc 2: Dịch vụ là tự trị. Các dịch vụ hoạt động như là các thực thể độc lập. Không có quyền làm chủ trong một môi trường hướng dịch vụ. Các dịch vụ được triển khai, thay đổi, quản lý một cách độc lập.

*Nguyên tắc 3:*Các dịch vụ chia sẻ giao diện và giao ước không chia sẻ cài đặt. Các dịch vụ tương tác với nhau chỉ dựa vào giao diện và giao ước sử dụng dịch vụ. Giao ước của dịch vụ mô tả cấu trúc của thông điệp và các ràng buộc giữa các thông điệp, điều này cho phép chúng ta bảo toàn được tính toàn vẹn của dịch vụ. Các giao ước và giao diện phải được duy trì tính ổn định với thời gian. Vì vậy việc xây dựng chúng một cách mềm dẻo là rất quan trọng.

Nguyên tắc 4: Tính tương thích của dịch vụ phải dựa trên chính sách. Cả người cung cấp và người dùng dịch vụ sẽ phải có các chính sách để tương tác qua các giao diện của dịch vụ. Một ví dụ đơn giản về chính sách phía người cung cấp là một dịch vụ có thể đòi hỏi người gọi phải có một tài khoản hợp lệ với người cung cấp dịch vụ. Về phía người dùng dịch vụ, một tổ chức có thể đòi hỏi các lời kêu gọi qua Internet phải được mã hoá.

Các quyết định thiết kế và kiến trúc trong kiến trúc hướng dịch vụ

Có rất nhiều quyết định thiết kế và kiến trúc là nền tảng cho SOA để đạt được các mục tiêu và lợi ích mà kiến trúc này đã nêu ra. Các quyết định kiến trúc thường liên quan tới việc chọn lựa các công nghệ và sản phẩm cần thiết kế để đáp ứng chất lượng cho các thuộc tính dịch vụ được yêu cầu bởi một quy trình nghiệp vụ. Các quyết định thiết kế tập trung vào các lựa chọn dịch vụ sử dụng kỹ thuật được mô tả sau trong phân tích và thiết kế hướng đối tượng. Một số các quyết định chính là:

1. Xác định dịch vụ: các quyết định thiết kế quyết định thành công một kiến trúc hướng dịch vụ.
2. Thiết kế cài đặt thành phần chứa: các quyết định thiết kế và kiến trúc rất quan trọng để một dịch vụ cung cấp các chức năng cho việc mở rộng và bảo trì.
3. Mức độ đóng gói : các lựa chọn thiết kế về dịch vụ phải phù hợp với mức độ tái sử dụng và mềm dẻo được yêu cầu trong ngữ cảnh cụ thể. Các dịch vụ có mức độ đóng gói cao hơn có thể trợ giúp việc đóng gói các thay đổi trong các dịch vụ kỹ thuật có mức độ đóng gói thấp hơn (các dịch vụ này thường có xu hướng thay đổi thường xuyên hơn so với các nghiệp vụ ở ở mức cao hơn). Các nhân tố của tính mềm dẻo sẽ là các tiêu chuẩn chính cho việc đóng gói hơn là việc chỉ đơn thuần đóng gói chức năng.

4. Tính gắn kết không chặt chẽ và cấu hình lại động là một quyết định thiết kế yêu cầu việc tách giao diện khỏi cài đặt giao thức thông qua các chuẩn mở. Kết nối giữa các thành phần cung cấp và sử dụng dịch vụ cần phải bền vững và được cấu trúc rõ ràng, có khả năng cấu hình lại linh hoạt. Có khả năng cấu hình lại có nghĩa là các thành phần sử dụng dịch vụ và các thành phần cung cấp dịch vụ hiện có thể được lắp ghép lại với các chức năng không bị thay đổi để đạt được các giải pháp nghiệp vụ trong các môi trường kỹ thuật khác nhau và với các ràng buộc thao tác khác nhau của các đối tác kinh doanh mới.

Các tầng SOA: Các thành phần của một kiến trúc hướng dịch vụ.

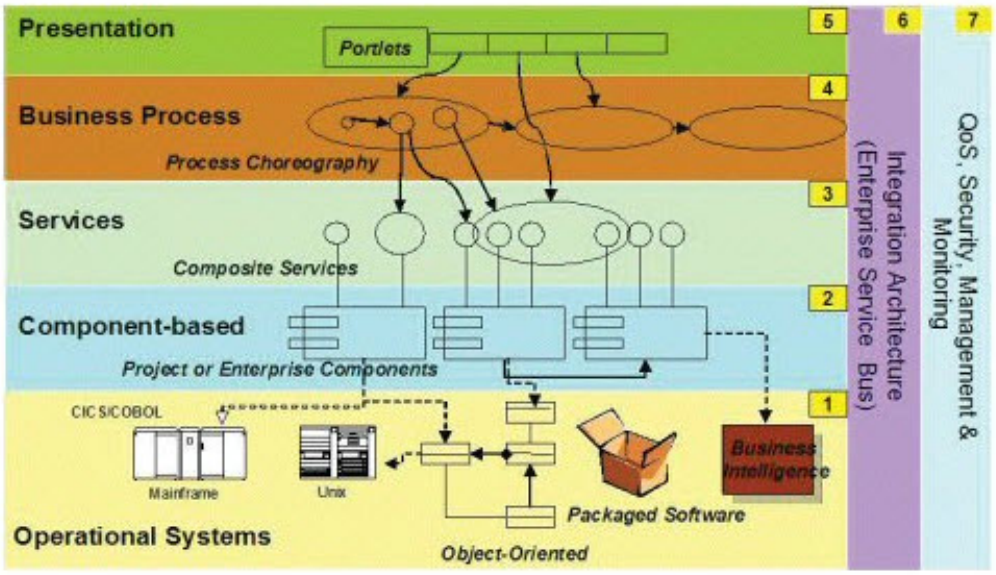
Danh mục dịch vụ: mô tả các dịch vụ nghiệp vụ trogn SOA, bao gồm một danh sách, phân loại và cấu trúc của các dịch vụ được xác định thông qua kỹ thuật phân tích và thiết kế hướng dịch vụ.

Các thành phần: cung cấp các cài đặt chức năng cuả dịch vụ.

Thành phần sử dụng dịch vụ, thành phần cung cấp dịch vụ và thành phần môi giới dịch vụ với các kho đăng ký dịch vụ, ở đó, các định nghĩa và mô tả dịch vụ được xuất bản.

Các tầng của SOA: vị trí của các thành phần và dịch vụ.

Hình 5.4 mô tả các tầng của SOA. Với mỗi tầng, các quyết định thiết kế và kiến trúc được tiến hành.



Các tầng của SOA.

Tầng 1: tầng dưới cùng, mô tả các hệ thống vận hành. Tầng này chứa các hệ thống hoặc các ứng dụng hiện có, bao gồm các ứng dụng được đóng gói, các ứng dụng đang có,

và các cài đặt hệ thống hướng đối tượng theo kiểu cũ cũng như các ứng dụng thu thập nghiệp vụ. Kiến trúc theo tầng của một kiến trúc hướng đối tượng có thể thúc đẩy các hệ thống hiện có, tích hợp chúng bằng cách sử dụng tích hợp hướng dịch vụ.

Tầng 2: tầng thành phần, sử dụng các kỹ thuật và thiết kế dựa đối tượng chứa trong phát triển hướng thành phần truyền thống.

Tầng 3: cung cấp các cơ chế để lấy các thành phần ở quy mô doanh nghiệp, các thành phần cho từng đơn vị doanh nghiệp cụ thể, và trong một số trường hợp là các thành phần của từng dự án và cung cấp các dịch vụ thông qua các giao diện của chúng. Trong tầng này, các giao diện được thể hiện ra ngoài thông qua các đặc tả dịch vụ, ở đó, các dịch vụ tồn tại một cách độc lập hoặc là tổng hợp của một số dịch vụ.

Tầng 4: là một sự tiến hoá của việc tổng hợp dịch vụ các luồng hoặc sắp đặt các nhiệm vụ đượcc phân nhóm vào một luồng để hoạt động như một ứng dụng. Các ứng dụng này hỗ trợ các trường hợp sử dụng các quy trình nghiệp vụ cụ thể. ở đây, các công cụ tổng hợp luồng trực quan có thể được sử dụng để thiết kế luồng ứng dụng.

Tầng 5: Tầng trình diễn thường nằm ngoài phạm vi của một kiến trúc hướng dịch vụ.

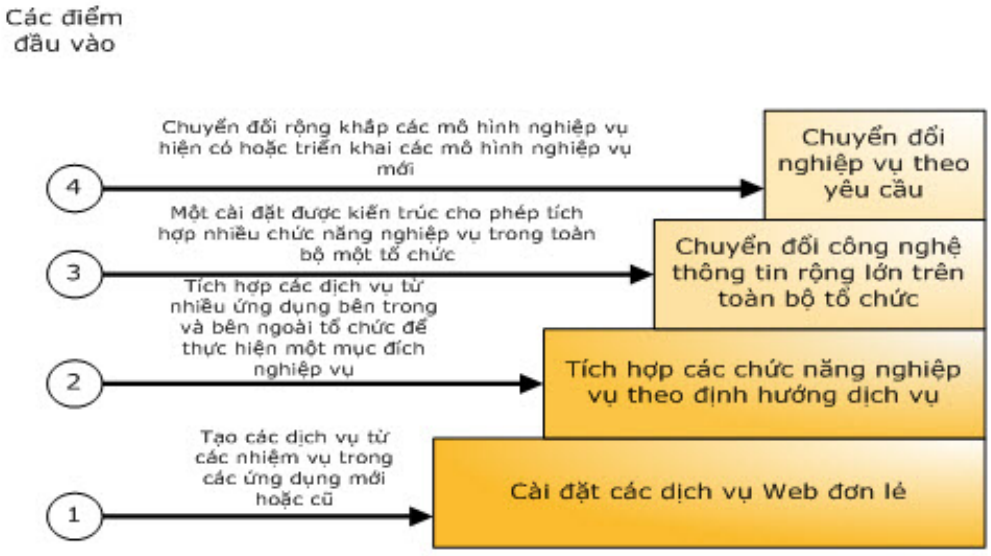
Tầng 6: cho phép sự tích hợp của các dịch vụ thông qua việc định tuyến tin cậy và thông minh, giao thức trung gian, và một số cơ chế chuyển đổi khác, thường đượcc mô tả như là một tuyến dịch vụ doanh nghiệp (ESB- Enterprise Services Bus).

Tầng 7: đảm bảo chất lượng dịch vụ thông qua các cơ chế cảm biến và đáp ứng các công cụ kiểm soát các ứng dụng SOA.

Các mức độ chấp nhận kiến trúc hướng dịch vụ

Việc sử dụng kiến trúc hướng dịch vụ không bị giới hạn cho các tổ chức hơn. Trong thực tế, kiến trúc này tạo ra cơ hội cho các tổ chức vừa và nhỏ. Một ví dụ về việc các doanh nghiệp nhỏ có thể sử dụng dịch vụ là việc nhận các hoá đơn: chúng ta có thể dùng dịch vụ mạng để tạo ra một dịch vụ nhận hoá đơn. Các hoá đơn này sẽ được lưu trữ bởi dịch vụ cho đến khi hệ thống kế toán trên máy PC của người dùng kết nối đến nó để nhận hoá đơn về bằng cách sử dụng dịch vụ mạng. Các hoá đơn sau đó có thể cập nhận một cách tự động trên PC… Bằng cách này, bất kỳ tổ chức nào, không phụ thuộc vào quy mô lớn hay nhỏ, đều có thể nhận được lợi ích từ việc sử dụng kiến trúc hướng dịch vụ.

Một tổ chức có thể dùng nhiều cách khác nhau để tiếp nhận được kiến trúc hướng dịch vụ, tuỳ thuộc vào mục tiêu và các ràng buộc công nghệ thông tin.



Các mức độ thực hiện kiến trúc hướng dịch vụ

Mức độ 1: Cài đặt các dịch vụ riếng lẻ.

Mức độ này tạo ra các dịch vụ từ các nhiệm vụ có trong các ứng dụng mới hoặc các ứng dụng hiện có. Kiến trúc hướng dịch vụ đem lại khả năng thiết kế các ứng dụng và các hệ thống cung cấp dịch vụ cho các ứng dụng khác thông qua các giao diện được xuất bản. Cách tạo ra các ứng dụng này có thể đưa đến một mô hình lập trình manh hơn, linh hoạt hơn, giảm chi phí cả trong phát triển và bảo trì.

Mức độ 2: Tích hợp hướng dịch vụ các chức năng nghiệp vụ

Mức độ tiếp theo là tích hợp hướng dịch vụ. Bước này bao gồm việc tích hợp các dịch vụ từ nhiều ứng dụng khác nhau cả bên trong lẫn bên ngoài doanh nghiệp để phục vụ cho mục đích nghiệp vụ. Mức độ này cũng phảI hỗ trợ nhiều kiểu tích hợp, bao gồm:

Tích hợp ứng dụng: gồm cả sự phát triển các giao diện mới để thể hiện các ứng dụng hiện có.

Tích hợp thông tin: gồm cả dữ liệu doanh nghiệp và dữ liệu phòng ban.

Tích hợp thông tin : gồm tích hợp qua sự tích hợp, sắp xếp các ứng dụng và dịch vụ với nhiều giao diện.

Tích hợp hệ thống: gồm sự kết nối các hệ thống không đồng nhất, các hệ thống hiện có và tích hợp ứng dụng.

Mức độ 3: Chuyển đổi doanh nghiệp công nghệ thông tin có quy mô lớn.

Mức cài đặt SOA rộng hơn là một cài đặt được kiến trúc hoá cho phép tích hợp nhiều chức năng nghiệp vụ trogn toàn doanh nghiệp.

Mức độ 4: Chuyển đổi nghiệp vụ theo yêu cầu.

Đây là mức độ cuối cùng trong phát triển theo kiến trúc hướng dịch vụ. Mức độ này là sự định hướng chiến lược hướng tới sự chuyển đổi rộng của các mô hình nghiệp vụ hiện có hoặc sự triển khai của các mô hình nghiệp vụ mới.

Các bước trong quy trình phát triển phần mềm theo định hướng dịch vụ

Hiện nay chưa có một quy trình cụ thể để phát triển các ứng dụng theo kiến trúc hướng dịch vụ, tuy nhiên, dựa trên thực tế, 12 bước sau đã được đưa ra nhằm tham khảo quyết định chuyển sang hướng dịch vụ.

Mười hai bước trong quy trình phát triển phần mềm theo kiến trúc hướng dịch vụ:

Hiểu nghiệp vụ.

Xác định phạm vi (miền) của vấn đề.

Hiểu tất cả ngữ nghĩa của ứng dụng trong miền đó.

Hiểu tất cả các dịch vụ hiện có trong miền.

Hiểu tất cả các nguồn và đích của thông tin có trong miền.

Hiểu tất cả các quy trình trong miền.

Xác định và phân loại tất cả các giao diện bên ngoài miền cần thiết cho việc xây dựng ứng dụng (các dịch vụ và thông tin).

Định nghĩa các dịch vụ mới và các ràng buộc thông tin của các ứng dụng dịch vụ đó.

Định nghĩa các dịch vụ mới, cũng như các dịch vụ và ràng buộc thông tin cho các quy trình này.

Lựa chọn tập công nghệ.

Triển khai công nghệ SOA.

Kiểm thử và đánh giá.

Bước 1: Hiểu các mục đích nghiệp vụ, và xác định thành công.

Đây là nhiệm vụ thu thập các yêu cầu cơ bản. Nó đòi hỏi phảI tiếp xúc với tài liệu, nhân sự và hệ thống để xác định thông tin cho phép việc tích hợp ứng dụng được xác định đúng để có thể phân tích, mô hình hoá và cải tiến. Chỉ sau khi thực hiện bước này thì thực thu tập giảI pháp thích hợp mới được ra đời.

Cần lưu ý rằng có cả lợi ích hữu hình và vô hình từ việc cài đặt bất kỳ loại công nghệ nào. Lợi ích hữu hình bao gồm việc tiết kiệm chi phí tức thì, như việc tự động hoá một hệ thống bán theo đơn đặt hàng thay cho việc bán hàng thủ công. Lợi ích vô hình thì khó nhận biết hơn, như sự thoả mãn của khách hàng dẫn tới việc tăng doanh số bán hàng, hoặc cộng tác chặt chẽ hơn tăng cường chất lượng và cho phép cac nhân công trao đổi thông tin tốt hơn.

Bước 2: Xác định miền vấn đề.

Phải xác định phạm vi của việc ứng dụng SOA trong một tổ chức. Hãy chia nhỏ tổ chức để áp dụng SOA thay vì áp dụng vào toàn bộ tổ chức. Cùng với thời gian, những thành công nhỏ sẽ dẫn tới các chiến lược thành công lớn hơn, phải thiết lập các đường ranh giới khi bắt đầu dự án có thể tiến hành xây dựng ứng dụng một cách trọng tâm hơn.

Bước 3: Hiểu tất cả các ngữ nghĩa ứng dụng trong miền vấn đề.

Chúng ta không thể giải quyết các vấn đề mà bản thân mình không hiểu rõ. Vì vậy, bước tiếp theo này là cực kỳ quan trọng để xác định tất cả các siêu dữ liệu ngữ nghĩa tồn tại trong miền ứng dụng nhằm giải quyết các vấn đề một cách hoàn hảo. Sự hiểu biết về ngữ nghĩa ứng dụng thiết lập cách thức và khuôn dạng trong đó ứng dụng tham khảo cac thuộc tính của quy trình nghiệp vụ. Ví dụ, cùng một số hiệu khách hàng nhưng trong một ứng dụng này có một giá trị và ý nghĩa hoàn toàn khác trong một ứng dụng khác. Việc hiểu ngữ nghĩa của ứng dụng đảm bảo rằng sẽ không có bất kỳ sự xung đột thông tin nào khi nó được tích hợp với các ứng dụng khác ở mức độ thông tin hoặc dịch vụ. Xác định ngữ nghĩa của ứng dụng là một công việc khó khăn vì có thể nhiều hệ thống mà chúng ta đang phân tích đã cũ hoặc mang tính độc quyền. Bước đầu tiên trong việc xác định và định vị ngữ nghĩa là tạo ra một danh sách các hệ thống ứng viên. Danh sách này sẽ cho phép chúng ta có thể xác định những kho chứa dữ liệu nào tồn tại trong các hệ thống đó. Bất kỳ công nghệ nào có thể xây dựng lại các lược đồ dữ liệu vật lý và logic cũng sẽ có ích trong việc xác định dữ liệu trong các miền vấn đề. Tuy nhiên, trong khi lược đồ và mô hình dữ liệu có thể cho phép nhìn vào cấu trúc của cơ sở dữ liệu thì chúng lại không thể xác định những thông tin đó được sử dụng như thế nào trong ngữ cảnh của dịch vụ hoặc ứng dụng; đó là lý do chúng ta cần tới các bước tiếp theo, Bằng việc hiểu rõ các ngữ nghĩa của ứng dụng, và đảm bảo rằng tất cả các hệ thống nguồn và đích trong và giữa các tổ chức được tích hợp một cách hoàn hảo.

Bước 4: Hiểu tất cả các dịch vụ hiện có trong miền.

Tìm hiểu các thông tin về dịch vụ, bao gồm:

1. Vị trí của dịch vụ.
2. Mục đích của dịch vụ.
3. Thông tin ràng buộc của dịch vụ.
4. Các phụ thuộc (nếu đó là một dịch vụ phức hợp).
5. Các vấn đề bảo mật.

Vị trí tốt nhất để bắt đầu tìm hiểu là thư mục dịch vụ. Giống như các thư mục khác, đây là một kho chứa các thông tin được thu nhập về các dịch vụ hiện có, cùng với các tài liệu về mỗi dịch vụ, bao gồm mục đích của dịch vụ, các thông tin vào/ra.. Thư mục này được sử dụng cùng với những hiểu biết về ngữ nghĩa của ứng dụng để xác định các điểm tích hợp trong tất cả các hệ thống của miền vấn đề.

Bước 5: Hiểu tất cả các nguồn và đích thông tin hiện có trong miền vấn đề.

Bước này xác định các giao diện xử lý các thông tin đơn giản. Chúng có thể thực hiện một trong hai vai trò: sử dụng thông tin (đích) hoặc cung cấp thông tin (nguồn).

Chúng ta cần phải hiểu rõ khía cạnh sau:

Vị trí của chúng.

Cấu trúc của luồng thông tin vào/ra.

Các ràng buộc tích hợp.

Các phụ thuộc (các nguồn và đích khác, cũng có thể là các dịch vụ).

Các vấn đề bảo mật.

Bước 6: Hiểu tất cả các quy trình.

Chúng ta cần xác định và liệt kê tất cả các quy trình nghiệp vụ tồn tại trong miền vấn đề, có thể là tự động hoá hoặc không phải là tự động hoá. Việc này rất quan trọng vì chúng ta đã biết dịch các dịch vụ và nguồn/đích thông nào hiện có, chúng ta cần xác định các cơ chế tương tác cao hơn, bao gồm tất cả các quy trình ở mức độ cao, mức trung bình và mức thấp. Trong nhiều trường hợp, những quy trình này vẫn chưa được tự động hoá hoặc chỉ có một phần được tự động hóa. Ví dụ, nếu một kiến trúc sư tích hợp ứng dụng cần hiểu tất cả các quy trình hiện có trong một ứng dụng kiểm kê, anh ta sẽ hoặc là đọc tài liệu hoặc là đọc mã nguồn để xác định quy trình nào đang được thực hiện. Sau đó, anh ta sẽ đưa quy trình nghiệp vụ vào phân loại và xác định mục đích của quy trình, ai là người sở hữu nó, nó chính xác là gì, và công nghệ thực hiện nó (Java hoặc C++...). Những quy trình này sau đó được gán với các quy trình mới để đáp ứng được các yêu

cầu nghiệp vụ. Chúng ta cần phải xem xét khái niệm quy trình chia sẻ và quy trình riêng. Một số quy trình là quy trình riêng, và do đó, chúng không chia sẻ với các thực thể bên ngoài(trong một số trường hợp, chúng thậm chí còn không chia sẻ với các phần khác của tôt chức). Các quy trình chia sẻ và quy trình riêng có thể tồn tại trong cùng một không gian quy trình với công nghệ tích hợp quy trình quản lý bảo mật giữa các người dùng.

Một số thông tin có thể được bảo trì trong phân loại, đó là thông tin bao gồm các biến được sử dụng trong các quy trình, các lược đồ đối tượng, các yêu cầu bảo mật, và/ hoặc các đặc điểm hiệu suất. Mỗi phân lịa quy trình phải duy trì tập thuộc tính riêng của nó, được xây dựng tuỳ biến cho mỗi yêu cầu tích hợp ứng dụng cụ thể.

Bước 7: Xác định và phân loại tất cả các giao diện bên ngoài miền cần thiết cho việc xây dựng ứng dụng.

Chúng ta cần xác định tất cả các giao diện bên ngoài mà các hệ thống trong miền của vấn đề của chúng ta có tương tác với hoặc cần tương tác với để đem lại giá trị tối đa. Điều quan trọng ở đây là phải chắc chắn rằng tất cả các giao diện cần thiết đều được xác định, bao gồm khả năng thể hiện các dịch vụ của miền đề ra bên ngoài cho các đối tác, cũng như khả năng nhận biết và thúc đẩy dịch vụ của họ. Các hệ thống của đối tác và của chúng ta cần hoạt động cùng nhau để hỗ trợ các quy trình chia sẻ chung.

Bước 8: Xác định các dịch vụ mới, các dịch vụ phức hợp và thông tin buộc đối với các dịch vụ đó.

Chúng ta cần xác định tất cả các dịch vụ tạo thành SOA; những dịch vụ này được chia làm 3 loại/

Bước 9: Xác định các quy trình mới, cũng như các dịch vụ và thông tin ràng buộc với các quy trình đó.

Đến bước này, chúng ta cần hiểu phần lớn những thành phần cần thiết để xác định các quy trình mới cũng như liên kết chúng với các quy trình hiện có tự động hoá các quy trình mà trước chưa được tự động hoá.

Bước 10: Lựa chọn tập công nghệ.

Có rất nhiều các công nghệ để lựa chọn, gồm các máy chủ ứng dụng, các đối tượng phân tán, và các máy chủ tích hợp. Sự lựa chọn công nghệ sẽ giống như một sự tổng hợp các sản phẩm và các nhà cung cấp để đáp ứng yêu cầu cho SOA. Rất hiếm có trường hợp một nhà cung cấp duy nhất có khả năng giải quyết được vấn đề.

Lựa chọn công nghệ là một công việc khó khăn yêu cầu một lượng thời gian và công sức đáng kể. Việc tạo ra tiêu chuẩn cho công nghệ và sản phẩm, việc hiểu rõ ràng các giải

pháp được đưa ra, và sau đó nối các tiêu chuẩn với các sản phẩm đó là một công việc không dễ dàng. Đê thành công, việc kết nối tiêu chuẩn với sản phẩm thường đòi hỏi một dự án thử nghiệm để chứng minh rằng nó sẽ hoạt động. Thời gian cần thiết để lựa chọn các công nghệ phù hợp có thể dài bằng thời gian phát triển SOA, nhưng nếu nản chí, có thể dẫn tới việc lựa chọn các công nghệ không phù hợp dẫn đến phá hỏng hệ thống.

Bước 11: Triển khai công nghệ SOA.

Đến bước này, chúng ta đã hiểu tất cả những gì cần thiết phải hiểu, đã xác định được các dịch vụ và quy trình mới, đã lựa chọn được tập công nghệ thích hợp, và bây giờ là thời gian để xây dựng hệ thống.

Bước 12: Kiểm thử và đánh giá.

Để đảm bảo cho việc kiểm thử cần xây dựng kế hoạch kiểm thử.

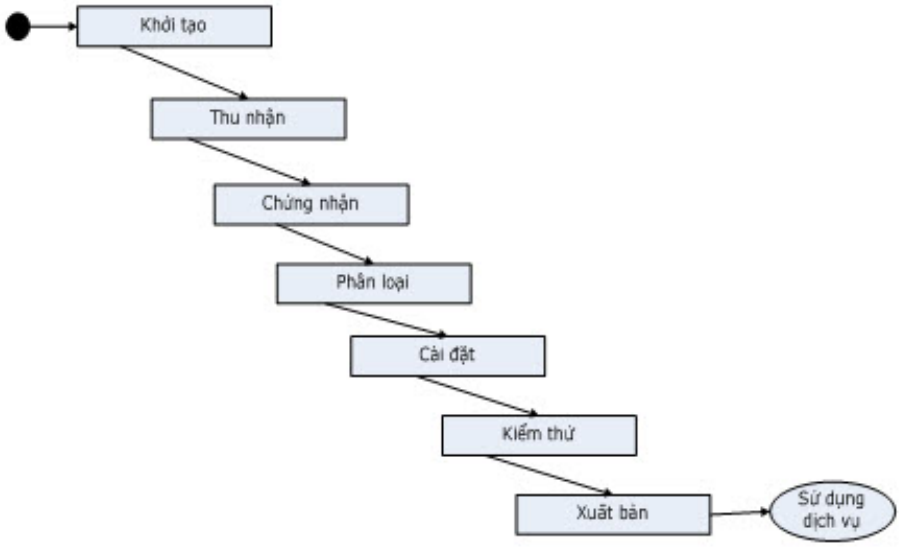
Cần phải hiểu rằng 12 bước trên không phải là quy trình bắt buộc xây dựng một dự án SOA thành công. Trong một số trường hợp, chúng ta cần thêm vào hoặc loại bỏ đi một số bước để phù hợp với yêu cầu cụ thể.

Kiến trúc hướng dịch vụ được đưa ra nhằm loại bỏ sự trùng lặp và dư thừa qua việc tái sử dụng và tích hợp. Cách đơn giản nhất là để bắt đầu với SOA là thử với một dịch vụ mà chúng ta biết rằng có nhiều cài đặt trong nhiều ứng dụng khác nhau, sau đó bắt đầu xây dựng kế hoạch và chiến lược để loại bỏ các dịch vụ dư thừa. Quy trình cài đặt này sẽ giúp chúng ta đáp ứng được các yêu cầu cơ bản của tổ chức ẩn sau bước chuyển đổi thành công sang SOA. Khi chúng ta đã thực hiện được quy trình này, chúng ta sẽ có các công cụ và hiểu biết để mở rộng quy mô áp dụng SOA trong tổ chức của mình.

Vòng đời phát triển của dịch vụ

Việc đảm bảo rằng chỉ có các dịch vụ nghiệp vụ chuẩn hoá và có mức trừu tượng cao trong xây dựng và triển khai là mối quan tâm chính trong việc xây dựng các dịch vụ trong kiến trúc hướng dịch vụ. Chỉ với việc tập trung chú ý vào quy trình xác định dịch vụ mới đảm bảo việc cài đặt SOA trở lên hiên hiệu quả như mong đợi.

Dưới đây là một vòng đời phát triển và định nghĩa dịch vụ cho phép tiến hành thực hiện việc xem xét và các điểm kiểm tra từ sớm trong quy trình định nghĩa dịch vụ. Điều này sẽ giúp loại bỏ lỗi trước khi chúng làm cho chi phí khắc phục lỗi tăng vọt khi dịch vụ được tiến hành phát triển và triển khai.



Vòng đời phát triển dịch vụ

Hình trên mô tả vòng đời phát triển dịch vụ, mỗi trạng thái được định nghĩa như sau:

Khởi tạo: một dịch vụ tiềm năng được xác định từ việc phân tích các yêu cầu của doanh nghiệp từ trên xuống (top – down) và việc mô hình hoá quy trình nghiệp vụ.

Thu nhận: đội ngũ kiến trúc dịch vụ thông báo là đã nhận được yêu cầu dịch vụ và bắt đầu đánh giá nó.

Chứng nhận: đội ngũ kiến trúc dịch vụ đã đánh giá dịch vụ và thống nhất giao diện chức năng của nó.

Phân loại: dịch vụ được xác định và được chuyển giao cho đội phát triển.

Cài đặt: đội ngũ phát triển dịch vụ cài đặt dịch vụ theo các quy tắc và hướng dẫn phát triển được định nghĩa bởi framework.

Kiểm thử: đội ngũ kiểm thử dịch vụ kiểm chứng tính năng cũng như các đặc tính chất lượng của dịch vụ.

Xuất bản: dịch vụ được xuất bản để có thể được sử dụng bởi các dịch vụ khác hoặc các ứng dụng định hướng quy trình.

Các công nghệ hướng dịch vụ

Sun JINI

Công nghệ Jini cho phép xây dựng một hệ thống là một mạng của các dịch vụ. Các dịch vụ có thể được thêm vào và xoá bỏ khỏi mạng, và người dùng mới có thể tìm kiếm các dịch vụ hiện có. Tất cả đều xảy ra động, không có sự quản lý.

Dịch vụ được dựa trên các giao diện đã được viết trong ngôn ngữ lập trình Java. Nó không quan tâm đến việc dịch vụ cài bằng phần cứng hay phần mềm. Đối tượng dịch vụ mà người dùng tải về được cung cấp bởi các thành phần cung cấp dịch vụ. Client chỉ cần biết rằng họ đang làm việc với một cài đặt của một giao diện được viết bằng ngôn ngữ lập trình Java. Việc thiết kế dựa trên các giao diện dịch vụ cho phép xây dựng các hệ thống với tính sẵn dùng cao. Một thành phần có thể sử dụng bất kỳ dịch vụ nào phù hợp với giao diện, thay vì được cấu hình tĩnh để giao tiếp với một thành phần nhất định nào đó.

Công nghệ Jini được xây dựng phía trên công nghệ Java (xem hình dưới). Phương thức triệu gọi từ xa của Java (RMI) cung cấp cơ chế thu rác từ xa của các đối tượng từ xa và khả năng chuyển trạng thái của đối tượng cũng như mã đối tượng qua mạng.

Mô hình kiến trúc Jini

Kiến trúc Jini bao gồm:

Dịch vụ tra cứu (Looup Services):

Dịch vụ tra cứu lưu trữ các dịch vụ Jini và cung cấp các uỷ nhiệm để truyền thông với dịch vụ, bản thân nó cũng là một dịch vụ Jini.

Dịch vụ Jini(Jini services):

Dịch vụ Jini được đăng ký với dịch vụ tra cứu và có khả năng được triệu gọi thông qua giao diện công khai của mình (giao diện này được định nghĩa thông qua một giao diện từ xa). Hệ thống nền tảng truyền thông của Jini là RMI.

Thành phần sử dụng dịch vụ Jini (Jini Client):

Thành phần sử dụng dịch vụ Jini là một phần mềm yêu cầu đối tượng uỷ nhiệm từ dịch vụ tra cứu để gọi dịch vụ Jini.

Jini có một số các dịch vụ được xây dựng sẵn, bao gồm:

Dịch vụ tìm kiếm tra cứu (Lookup Discovery Services):

Dịch vụ tìm kiếm tra cứu thông báo cho các thành phần sử dụng dịch vụ về các thay đổi trong mạng Jini. Các dịch vụ có thể kết hợp hoặc tách ra khỏi mạng bất kỳ thời điểm nào.

Dịch vụ tái tạo ràng buộc(Lease Renewal Services):

Khái niệm tái tạo ràng buộc hỗ trợ mạng dịch vụ Jini tính naưng tự hàn gắn và khắc phục lỗi. Dịch vụ phải tái tạo ràng buộc không được tái tạo, dịch vụ sẽ là một ứng viên bị loại bỏ khỏi mạng lưới dịch vụ. Trách nhiệm của những người quản trị trong lĩnh vực này được giảm tới mức tối thiểu nhờ tính năng tự hàn gắn của hệ thống.

Dịch vụ giao dịch (Transactor Services):

Dịch vụ cho phép sử dụng các giao dịch trong một hệ thống phân tán. Thông thường, các tổ chức sử dụng cơ sở dữ liệu để tạo các hệ thống giao dịch. Dịch vụ giao dịch của Jini đưa tính năng giao dịch của cơ sở dữ liệu lên mạng. Các dịch vụ có thể tham gia vào các giao dịch để đảm bảo các thuộc tính ACID (Atomicity- Consistency- Isolation- Durability) gắn liền với giao dịch.

Dịch vụ hộp thư sự kiện (Event MailBox Services):

Các thay đổi trong mạng dịch vụ Jini đươc truyền đi trogn hệ thống bằng cách sử dụng các sự kiện phân tán. dịch vụ hộp thư sự kiện hỗ trợ tính năng thông báo các sự sự kiện cho các dịch vụ ngay cả khi dịch vụ không được kích hoạt tại thời điểm hiện tại.

Openwings

Openwings là một khung kiến trúc hướng dịch vụ cho việc xây dựng các hệ thống các siêu hệ thống (hệ thống của hệ thống). Mặc dù không bị ràng buộc cụ thể với Jini, nó cho phép xây dựng dựa trên các khải niệm của Java và Jini để cung cấp một giải pháp hoàn thiện hơn.

Openwings có hàng loạt các dịch vụ cốt lõi hỗ trợ tính toán hướng đối tượng.

Component Services (Các dịch vụ thành phần): cung cấp các kỹ thuật cho việc đưa ra công bố một dịch vụ trong hệ thống đảm bảo cho quá trình tìm kiếm phát hiện (discover), thông qua sử dụng component cung các thư viện cho phép:

Tạo ra các khả năng cung cấp, định vị và sử dụng các dịch vụ nói chung trong hệ thống mà không phụ thuộc vào bất cứ kỹ thuật định vị/ tìm kiếm (location/lookup) nào. Ví dụ các kỹ thuật định vị/tìm kiếm như Jini, UDDI với mô hình Web-services, giao thức phát hiện (discovery) Bluetooth.

Quá trình giao tiếp giữa các thành phần hay dịch vụ được định nghĩa với các API chuẩn cho các dịch vụ thông qua việc thiết kế thừa từ những giao diện chuẩn mà openwings đề xuất.

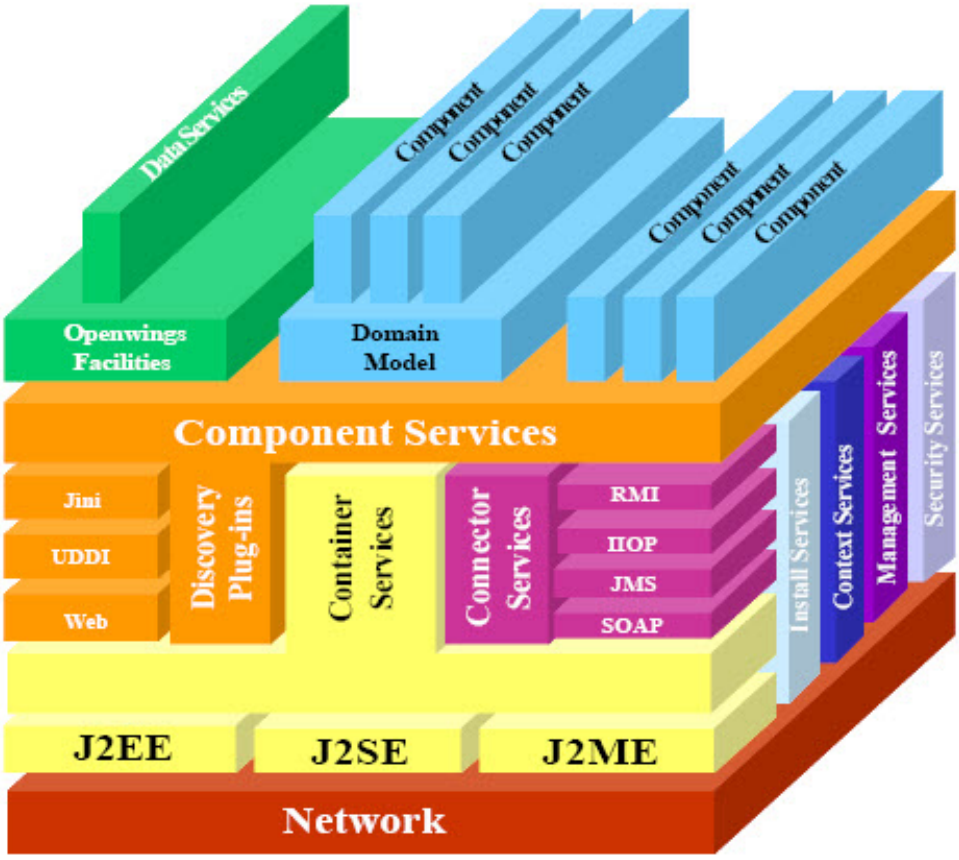
Cung cấp các component API cho giao thức tương tác giữa các dịch vụ qua mạng mà không phụ thuộc vào tính năng đồng bộ hay không đồng bộ của các dịch vụ.

Cung cấp khả năng điều khiển các dịch vụ thành phần ngay trong thời điểm dịch vụ đó đang được gọi thực thi.

Hỗ trợ việc đưa ra các báo cáo về trạng thái của các thành phần tham gia vào hệ thống.

Cho phép thiết lập môi trường thực tho động dựa theo yêu cầu tại thời điểm sử dụng.

Qua hình vẽ minh hoạ sau, ta có thể thấy được vị trí trung tâm của các dịch vụ thành phần trong mô hình khung mà Openwings đề xuất:



Mô hình khung của Openwings

Install services (Các dịch vụ cài đặt): là một thành phần dịch vụ của framework cho phép thực hiện cài đặt các thành phần mới vào hệ thống. Để cài đặt được thì các thành phần này cần phải thông qua các quy trình chuẩn mà openwings đề xuất.

Cung cấp các khả năng cài đặt thêm những dịch vụ mới để hạn chế tối đa các tương tác bằng tay cho người sử dụng. Đảm bảo không gây ảnh hưởng tới các thành phần hay dịch vụ đã được cài đặt trước đó.

Cung cấp các khả năng tự động dò tìm và cho phép gọi thực hiện với các thành phần được lưu trữ trong các thiết bị lưu trữ tự động.

Có cơ chế thẩm định quyền và độ ưu tiên cho việc cài đặt và gọi thực hiện.

Có cơ chế huỷ cài đặt an toàn cho hệ thông khi cần thiết.

Context Services (Các dịch vụ ngữ cảnh): là các thành phần cho hỗ trợ việc kết hợp các dịch vụ trong môi trường hệ thống để có được một dịch vụ lớn hơn. Đồng thời còn hỗ trợ việc kết hợp các dịch vụ được chia sẻ trong một mạng WAN.

Management Services (các dịch vụ quản lý): cung cấp các phương thức cơ bản hỗ trợ việc quản lý các thành phần và dịch vụ trong hệ thống. Với các hỗ trợ từ dạng quản lý hệ thống qua can thiệp bằng tay hay thông qua việc cài đặt các cơ chế quản lý tự động. Các hỗ trợ này được đóng gói trong Mbean.

Security Services (các dịch vụ kết bảo mật): cung cấp các phương thức cho việc mã hoá, truyền tải dữ liệu trong hệ thống, các hỗ trợ cơ bản cho việc cấp quyền, thẩm định quyền, đảm bảo an toàn khi truyền tin, tính toàn vẹn, khả năng phát hiện tấn công và đáp trả các tấn công vào hệ thống.

Connector Services (các dịch vụ kết nối): cung cấp các kỹ thuật cho việc giao tiếp giữa các thành phần hay dịch vụ trong hệ thống. Hỗ trợ trực tiếp cho các kỹ thuật giao tiếp thông qua một đối tượng chuyển tiếp trung gian như CORBA, RMI, giao tiếp từ một giao diện dịch vụ được định hướng trước theo các kỹ thuật chuyển tiếp trung gian chuẩn với các cơ chế động ngay tại thời điểm diễn ra giao tiếp tuỳ theo yêu cầu sử dụng. Với đầy đủ các hỗ trợ cho việc giao tiếp theo phiên hay theo dạng giao tiếp không duy trì kết nối. Hỗ trợ đầy đủ các hàm cơ bản cho phép làm việc với RMI, CORBA, JMS, SOAP...

Container Services (các dịch vụ chứa đựng): cung cấp môi trường tương ứng cho việc thực hiện các dịch vụ trong mô hình openwings. Đây là một thành phần quan trọng trong việc tạo nên khả năng vận hành động không phụ thuộc vào nền tảng hay môi trường vận hành phân tán của hệ thống.

Cung cấp các khả năng gọi chạy dịch vụ trên máy hiện tại qua một nền hệ thống từ xa.

Hỗ trợ việc quản lý vòng đời của các tiến trình dịch vụ, các phương thức cho phép tự động khởi động lại, tạm dừng hay huỷ dịch vụ.

Cung cấp khả năng gán hay thiết lập cho một số tiến trình xây dựng bằng Java có thể hoạt động độc lập trên các máy ảo Java mà không ảnh hưởng tới các thành phần khác.

Hỗ trợ việc gọi khởi động các dịch vụ không được xây dựng trên nền Java.

Cung cấp các phương thức cho ohép theo dõi tài nguyên hệ thống như theo dõi các thông tin về độ dung lượng trong bộ nhớ, khả năng hoạt động của bộ vi xử lý hay khả năng đáp ứng lưu lượng của đường truyền tải dữ liệu.

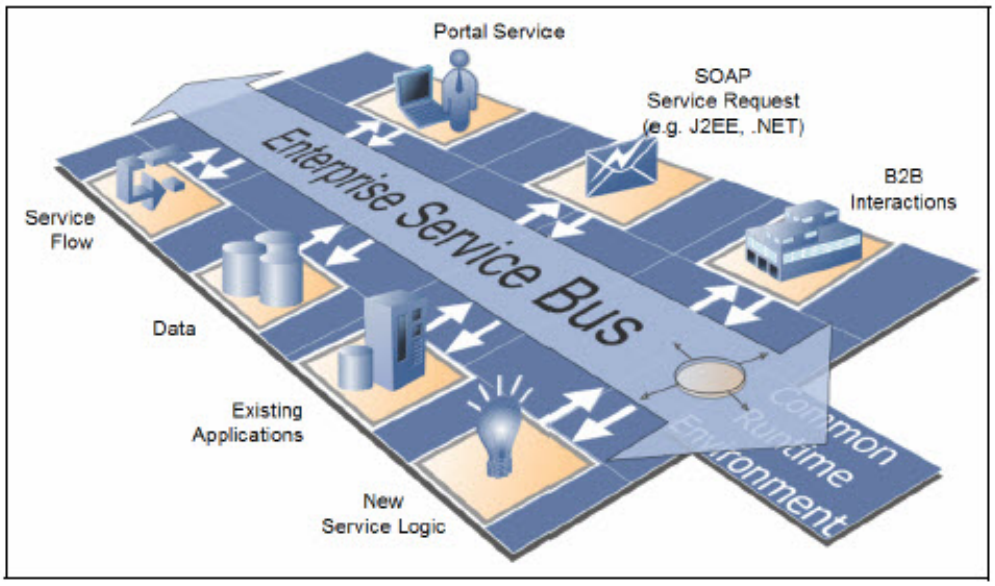
Dịch vụ web

Mặc dù các khái niệm nền tảng cho kiến trúc hướng dịch vụ đã được thiết lập từ trước khi dịch vụ mạng xuất hiện nhưng dịch vụ mạng vẫn đóng một vai trò quan trọng trong một kiến trúc hướng dịch vụ. Đó là bởi vì dịch vụ mạng được xây dựng trên một tập các giao thức được biết tới nhiều và độc lập về nền tảng. Các giao thức này bao gồm HTTP,

XML, UDDI, và SOAP. Chính sự kết hợp của các giao thức này đã làm dịch vụ mạng đáp ứng được các yêu cầu của chính kiến trúc hướng dịch vụ: SOA yêu cầu dịch vụ phải được phát hiện và triệu gọi động, yêu cầu này được thoả mãn bằng UDDI, WSDL, và SOAP; SOA yêu cầu dịch vụ có một giao ước giao diện độc lập nền tảng, yêu cầu này được thoả mãn bởi XML; SOA nhấn mạnh tới tính liên thông, yêu cầu này được thoả mãn bởi HTTP. Đây là lý do tại sao các dịch vụ mạng mang lại có vai trò trung tâm trong kiến trúc hướng dịch vụ. Chi tiết hơn về dịch vụ mạng sẽ được đề cập trong chương sau.

Enterprise Services Bus (ESB)

Các công nghệ dựa trên nền dịch vụ mạng ngày càng được ứng dụng rộng rãi trong phát triển và tích hợp ứng dụng trong các tổ chức. Một trong những vấn đề nổi lên hiện nay là việc tìm kiếm các phương pháp hiệu quả hơn trong việc thiết kế, phát triển và triển khai dịch vụ mạng dựa trên các hệ thống kinh doanh; quan trọng hơn nữa là việc chuyển kiểu truyền thông dịch vụ mạng điểm tới điểm tới các ứng dụng lớn hơn của các công nghệ này thành các quy trình kinh doanh ở mức xí nghiệp. Trong bối cảnh này, mô hình ESB đang xuất hiện như một bước tiến mới trong sự phát triển của dịch vụ mạng và kiến trúc hướng đối tượng.



Mô hình ESB

Dịch vụ mạng cơ bản:

Dịch vụ mạng cơ bản (SOAP/HTTP điểm tới điểm) cung cấp một nền tảng vững trắc cho việc cài đặt một kiến trúc hướng dịch vụ, nhưng có những vấn đề ảnh hưởng tới tính mềm dẻo và khả năng bảo trì của chúng trong các kiến trúc ở quy mô xí nghiệp.

Thứ nhất, bản chất điểm tới điểm của dịch vụ mạng cơ bản có nghĩa là người dùng dịch vụ thường cần phải được sửa đổi bất kỳ khi nào giao diện người cugn cấp dịch vụ thay đổi. Điều này không thành vấn đề trên quy mô nhỏ, nhưng trong các xí nghiệp lớn chúng ta sẽ phải phải thay đổi đối với các trình khách đã có.

Thứ hai chúng ta có thể kết thúc bằng một kiến trúc dễ đổ vỡ và không linh hoạt khi một số dung lượng lớn người dùng dịch vụ và người cung cấp dịch vụ liên lạc với nhau sử dụng các kết nối kiểu điểm tới điểm hỗn loạn.

Cuối cùng, dịch vụ mạng cơ bản đòi hỏi mỗi người dùng phải có một bộ điều hợp giao thức thích hợp với mỗi nhà cung cấp dịch vụ. Việc phải triển khai nhiều bộ điều hợp giao thức trên nhiều ứng dụng khách làm tăng giá thành và chi phí bảo trì.

Cách tiếp cậu ESB sẽ giải quyết được những vấn đề này.

Vậy ESB là gì ?

Khái niệm ESB không phải là một sản phẩm, nhưng là một thực tiễn kiến trúc tốt nhất cho việc cài đặt một kiến trúc hướng dịch vụ. Như chỉ ra trong hình dưới, nó thiết lập một đường bus thông điệp lớp xí nghiệp kết hợp hạ tầng thông điệp với sự chuyển đổi thông điệp và đinh tuyến dựa vào nội dung trong một tâng tích hợp logic giữa những người dùng và người cung cấp dịch vụ.

Mục đích của ESB là cung cấp một sự trừu tượng hoá về các nguồn tài nguyên của doanh nghiệp, cho phép các nghiệp vụ của doanh nghiệp có thể được phát triển và quản lý độc lập với hạ tầng, mạng và có sự có mặt của các dịch vụ doanh nghiệp khác. Các nguồn tài nguyên trong ESB được mô hình như những dịch vụ có một hay nhiều thao tác.

Cài đặt một ESB đòi hỏi một tập hợp được tích hợp của các dịch vụ phần giữa hỗ trợ các kiểu kiến trúc như sau:

Các kiến trúc hướng dịch vụ: ở đây, các ứng dụng phân tán bao gồm các dịch vụ có thể sử dụng lại được với các giao diện rõ ràng, có thể được xuất bản và tương thích với các chuẩn.

Các kiến trúc hướng thông điệp: ở đây, các ứng dụng gửi thông điệp qua ESB tới các ứng dụng nhận thông điệp.

Các kiến trúc hướng sự kiện: ở đây, các ứng dụng tạo mới và sử dụng các thông điệp độc lập lẫn nhau.

Các dịch vụ phần giữa(middleware): được cung cấp bởi một ESB cần chứa:

Phần giữa truyền thông hỗ trợ nhiều mô hình truyền thông (như đồng bộ, không đồng bộ, yêu cầu/ trả lời, một chiều...), chất lượng dịch vụ (bảo mật, hiệu năng...), các API, nền tảng và các giao thức độc lập.

Một cơ chế cho việc chèn xử lý thông minh của các lần yêu cầu và đáp ứng dịch vụ trong mạng.

Các công cụ dựa theo chuẩn để cho phép sự tích hợp nhanh chóng của các dịch vụ.

Hệ thống quản lý cho các ứng dụng liên kết lỏng và các tương tác của chúng.

Kiến trúc hướng dịch vụ xác định một kiến trúc hướng dịch vụ trừu tượng nhằm mục đích xây dựng các hệ thốn phần mềm bằng cách liên kết và kết tập các dịch vụ cục bộ và từ xa một cách mềm dẻo. Trong khi các mô hình kiến trúc trước đó kiểm soát tính phức tạp bằng cách gom nhóm các chức chung, SOA lại cố gắng thực hiện việc xác định các yêu cầu kiến trúc cụ thể đảm bảo rằng các công nghệ hỗ trợ đảm nhiệm trách nhiệm công nghệ chính. Bằng cách này, SOA đạt được một kiểu kiến trúc trừu tượng mà tập trung chính vào việc lắp ráp các hoạt động nghiệp vụ theo các yêu cầu. Công nghệ dịch vụ mạng hiện tại đang là một công nghệ hỗ trợ nổi bật nhất cho SOA. Nó cung cấp khả các giải pháp kỹ thuật cho phép thực hiện hoá các thành phần của hệ thống phần mềm theo kiến trúc hướng dịch vụ. Với sự chú trọng tập trung vào việc thiết kế các quy trình nghiệp vụ, SOA yêu cầu việc phát triển phần mềm tương tác chặt chẽ với môi trường nghiệp vụ. Một sự tham gia tích cực của các nhà quản lý và phân tích nghiệp vụ có thể cải tiến một cách đáng kể kết quả của việc thiết kế hệ thống.