**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP

TÌM HIỂU ARCHIMATE VÀ XÂY DỰNG HỆ THỐNG QUẢN LÝ TRƯỜNG TRUNG HỌC PHỔ THÔNG

**Ngành : Công nghệ thông tin**

**Niên khoá : 2006-2010**

**Lớp : DH06DTM**

**Sinh viên thực hiện : Nguyễn Duy Chinh**

**Nguyễn Hải Đăng**

**Nguyễn Tấn Mơ**

**Lê Hữu Tài**

TP.HỒ CHÍ MINH, tháng 9 năm 2010

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP

TÌM HIỂU ARCHIMATE VÀ XÂY DỰNG HỆ THỐNG QUẢN LÝ TRƯỜNG TRUNG HỌC PHỔ THÔNG

**Giáo viên hướng dẫn: Sinh viên thực hiện:**

**ThS. Nguyễn Đức Công Song Nguyễn Duy Chinh**

**Nguyễn Hải Đăng**

**Nguyễn Tấn Mơ**

**Lê Hữu Tài**

TP.HỒ CHÍ MINH, tháng 9 năm 2010

CÔNG TRÌNH HOÀN TẤT TẠI

TRƯỜNG ĐẠI HỌC NÔNG LÂM TP. HCM

🙞🏵🙜

Giáo viên hướng dẫn: **Nguyễn Đức Công Song**

Giáo viên phản biện: **Nguyễn Thanh Phước**

Luận văn tốt nghiệp được báo cáo tại KHOA CÔNG NGHỆ THÔNG TIN TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM, ngày … tháng … năm 2010.

*Nhận xét của giáo viên hướng dẫn*

*Nhận xét của giáo viên phản biện*

**SINH VIÊN THỰC HIỆN:**

# LỜI CẢM ƠN

**🙞🏵🙜**

*Sau thời gian nghiên cứu đồ án, chúng em cũng đã đạt được những kết quả nhất định. Để đạt được điều này thì ngoài sự cố gắng và nổ lực của từng thành viên trong nhóm, chúng em còn nhận được rất nhiều sự quan tâm và chỉ bảo của nhà trường, quý thầy cô, gia đình, bạn bè, …*

*Chúng em xin chân thành cám ơn Khoa Công nghệ thông tin Đại học Nông Lâm Thành phố Hồ Chí Minh đã tạo điều kiện cho chúng em thực hiện đề tài này.*

*Chúng em chân thành cám ơn quý thầy cô là những người đã tận tình chỉ bảo và truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian qua.*

*Chúng em xin chân thành biết ơn Thầy Nguyễn Đức Công Song đã tận tình hướng dẫn, chỉ bảo và giúp đỡ chúng em trong suốt quá trình thực hiện đề tài nghiên cứu này.*

*Ngoài ra chúng em còn xin gửi lời cảm ơn tới nhà Trường, văn phòng Khoa Công nghệ thông tin và bạn bè là những người đã chân thành giúp đỡ chúng em trong thời gian qua.*

*Trong quá trình thực hiện đề tài nghiên cứu, mặc dù các thành viên đã cố gắng nỗ lực thực hiện nhưng chúng em chắc không thể tránh được những sai sót nhất định. Kính mong sự thông cảm và tận tình chỉ bảo của quý Thầy Cô.*

*Sinh viên thực hiện*

*Nguyễn Duy Chinh*

*Nguyễn Hải Đăng*

*Nguyễn Tấn Mơ*

*Lê Hữu Tài*

# DANH SÁCH CHỮ VIẾT TẮT

**UML U**nified **M**odel **L**anguage

Ngôn ngữ mô hình hợp nhất

**SOA** **S**ervice **O**riented **A**rchitecture

Kiến trúc hướng dịch vụ

**EA** **E**nterprise**A**rchitecture

Kiến trúc Enterprise

# DANH SÁCH CÁC THUẬT NGỮ TIẾNG ANH

**Business process** quy trình nghiệp vụ

**Actor** ngườiliên quan đến hệ thống

**Module** môđun

**Domain experts** là người có kiến thức chuyên môn hay các kỉ năng trong một lĩnh vực riêng biệt.

**Domain architect** kiến trúc sư về một lĩnh vực nào đó

**Process architect** kiến trúc sư của một quy trình nghiệp vụ

**Application architect** kiến trúc sư của một ứng dụng

**ArchiMate shape** những ký hiệu dùng trong ngôn ngữ archimate

**Visualization** thể hiện của view qua những hình ảnh sinh động dễ hiểu

**View** bản vẽ thể hiện kiến trúc

**Architecture description** bản mô tả thể hiện đầy đủ những thông tin về kiến trúc

**Specialization** sự kế thừa và mở rộng

**High level** cấp độ trừu tượng hoá

**Internal service** dịch vụ hướng cung cấp cho những thành phần bên trong

**External service** dịch vụ hướng cung cấp ra bên ngoài

# MỤC LỤC

🙞🏵🙜

[MỤC LỤC iv](#_Toc272438359)

[DANH MỤC CÁC HÌNH vi](#_Toc272438360)

[TÓM TẮT viii](#_Toc272438361)

[Tên đề tài viii](#_Toc272438362)

[Nội dung nghiên cứu viii](#_Toc272438363)

[Hướng tiếp cận và giải quyết vấn đề viii](#_Toc272438364)

[Một số kết quả đạt được ix](#_Toc272438365)

[CHƯƠNG 1. MỞ ĐẦU 1](#_Toc272438366)

[1.1. LÝ DO CHỌN ĐỀ TÀI 1](#_Toc272438367)

[1.2. MỤC TIÊU ĐỀ TÀI 1](#_Toc272438368)

[1.3. PHẠM VI NGHIÊN CỨU 1](#_Toc272438369)

[CHƯƠNG 2. TỔNG QUAN 3](#_Toc272438370)

[2.1. ĐẶT VẤN ĐỀ 3](#_Toc272438371)

[CHƯƠNG 3. NỘI DUNG NGHIÊN CỨU 4](#_Toc272438372)

[**3.1. Giới thiệu cơ bản về Enterprise Architecture 4**](#_Toc272438373)

[3.1.1. Các thuật ngữ chuyên môn 4](#_Toc272438374)

[3.1.2. Tại sao phải có kiến trúc enterprise 5](#_Toc272438375)

[3.1.3. Kiến trúc sư enterprise – EA 5](#_Toc272438376)

[3.1.4. Quy trình kiến trúc hệ thống 6](#_Toc272438377)

[3.1.5. Sự truyền thông trong quá trình kiến trúc Enterprise 7](#_Toc272438378)

[3.1.6. Các phương pháp kiến trúc 11](#_Toc272438379)

[3.1.7. Các framework hỗ trợ 11](#_Toc272438380)

[3.1.8. Các ngôn ngữ kiến trúc hệ thống 13](#_Toc272438381)

[**3.2. Archimate language 18**](#_Toc272438382)

[3.2.1. ArchiMate là gì 18](#_Toc272438383)

[3.2.2. Tại sao dùng ArchiMate 19](#_Toc272438384)

[3.2.3. Những lợi ích của ArchiMate 20](#_Toc272438385)

[3.2.4. Các khái niệm chính trong ArchiMate 20](#_Toc272438386)

[3.2.5. Những ký hiệu của ArchiMate 21](#_Toc272438387)

[3.2.6. Kiến trúc tổng quát ngôn ngữ Archimate 24](#_Toc272438388)

[3.2.7. Tầng Nghiệp Vụ 27](#_Toc272438389)

[3.2.8. Tầng Ứng Dụng 37](#_Toc272438390)

[3.2.9. Tầng Kỹ thuật 42](#_Toc272438391)

[3.2.10. Viewpoint 46](#_Toc272438392)

[3.2.11. ArchiMate Viewpoint 55](#_Toc272438393)

[3.2.12. Viewpoint Framework 81](#_Toc272438394)

[3.2.13. Archimate và UML 83](#_Toc272438395)

[**3.3. SOA 83**](#_Toc272438396)

[3.3.1. Giới thiệu tổng quan 83](#_Toc272438397)

[3.3.2. Lịch sử SOA 84](#_Toc272438398)

[3.3.3. Tại sao dùng SOA 84](#_Toc272438399)

[3.3.4. SOA giải quyết 4 bài toán 84](#_Toc272438400)

[3.3.5. Vậy SOA là gì ? 84](#_Toc272438401)

[3.3.6. SOA không phải là một kỹ thuật 85](#_Toc272438402)

[3.3.7. Dịch vụ trong SOA 85](#_Toc272438403)

[3.3.8. Định nghĩa “Dịch vụ” trong SOA 85](#_Toc272438404)

[3.3.9. SOA tách riêng thực hiện thực dịch vụ với giao tiếp gọi dịch vụ. 85](#_Toc272438405)

[3.3.10. Ưu điểm 86](#_Toc272438406)

[3.3.11. Hai nguyên tắc thiết kế SOA 86](#_Toc272438407)

[3.3.12. Triển khai SOA cho doanh nghiệp 86](#_Toc272438408)

[3.3.13. Giai đoạn định nghĩa SOA 87](#_Toc272438409)

[**3.4. Open ESB 87**](#_Toc272438410)

[3.4.1. Giới thiệu 87](#_Toc272438411)

[3.4.2. GlassFish ESB 87](#_Toc272438412)

[3.4.3. Tại sao dùng ESB để hiện thực SOA 88](#_Toc272438413)

[3.4.4. Các thành phần “out of the box” mà OpenESB hổ trợ. 88](#_Toc272438414)

[3.4.5. BPEL - Business Process Execution Language 88](#_Toc272438415)

[3.4.6. Ta có thể làm gì với BPEL 89](#_Toc272438416)

[3.4.7. Cài đặt 90](#_Toc272438417)

[3.4.8. HelloWord project 90](#_Toc272438418)

[3.4.9. Kết nối database với Database BC 109](#_Toc272438419)

[3.5. Bài toán ứng dụng 124](#_Toc272438420)

[3.5.1. Phát biểu bài toán 124](#_Toc272438421)

[3.5.2. Các chức năng cần xây dựng 124](#_Toc272438422)

[3.5.3. Mô hình kiến trúc thể hiện qua archimate 126](#_Toc272438423)

[CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN 136](#_Toc272438424)

[4.1. KẾT QUẢ ĐẠT ĐƯỢC 136](#_Toc272438425)

[4.2. HƯỚNG PHÁT TRIỂN 136](#_Toc272438426)

[TÀI LIỆU THAM KHẢO 137](#_Toc272438427)

# DANH MỤC CÁC HÌNH

**Hình** **Trang**

[Hình 3-1 Vòng đời của quy trình kiến trúc hệ thống 14](#_Toc263679670)

[Hình 3-2 Sự truyền thông giữa Enterprise Architects với các Stakeholders 15](#_Toc263679671)

[Hình 3-3 Ví dụ về một chuỗi các hội thảo 16](#_Toc263679672)

[Hình 3-4 Vòng đời phát triển kiến trúc của phương pháp TOGAF 19](#_Toc263679673)

[Hình 3-5 Zachman Framework (Zachman 1987) 19](#_Toc263679674)

[Hình 3-6 TOGAF (based on The Open Group 2002). 20](#_Toc263679675)

[Hình 3-7 MDA framework. 20](#_Toc263679676)

[Hình 3-8 IDEF0 representation. 21](#_Toc263679677)

[Hình 3-9 Ví dụ mô hình được thể hiện bởi BPMN 22](#_Toc263679678)

[Hình 3-10 Mô hình quy trình nghiệp vụ trong Testbed. 22](#_Toc263679679)

[Hình 3-11 Quy trình thanh toán hóa đơn thể hiện bởi ARIS. 23](#_Toc263679680)

[Hình 3-12 Ví dụ một mô hình UML 25](#_Toc263679681)

[Hình 3-13 Các miền trong ArchiMate 25](#_Toc263679682)

[Hình 3-14 Các khái niệm chính trong ArchiMate 27](#_Toc263679683)

[Hình 3-15 Các khái niệm trong ArchiMate 28](#_Toc263679684)

[Hình 3-16 Những ký hiệu đối tượng 29](#_Toc263679685)

[Hình 3-17 Tổng quan về các khái niệm ArchiMate và mối quan hệ chính. 32](#_Toc263679686)

[Hình 3-18 Mô hình phân tầng ArchiMate 33](#_Toc263679687)

[Hình 3-19 Meta model của tầng nghiệp vụ 34](#_Toc263679688)

[Hình 3-20 Meta model của tầng ứng dụng 45](#_Toc263679689)

[Hình 3-21 Metamodel Tầng kỹ thuật 50](#_Toc263679690)

[Hình 3-22 Mô hình khái niệm về architecture description (IEEE Computer Society, 2000) 54](#_Toc263679691)

[Hình 3-23 Mối quan hệ giữa Model-view-visualization 56](#_Toc263679692)

[Hình 3-24 Minh hoạ của visualization từ những “archimate shape” 56](#_Toc263679693)

[Hình 3-25 Lanscape map minh hoạ cho hệ thống bảo hiểm ArchiSurance 57](#_Toc263679694)

[Hình 3-26 Phân loại viewpoint theo *mục đích và nội dung (*ArchiMate D3.4.1a v2.6) 60](#_Toc263679695)

[Hình 3-27 Hướng dẫn thiết kế viewpoint(ArchiMate D3.4.1a v2.6) 62](#_Toc263679696)

[Hình 3-28 Các ArchiMate viewpoint cơ bản 62](#_Toc263679697)

[Hình 3-29 Những khái niệm chính được sử dụng trong Introductory Viewpoint 63](#_Toc263679698)

[Hình 3-30 Những khái niệm chính được dung trong Organization Viewpoint: 64](#_Toc263679699)

[Hình 3-31 Ví dụ về Organization Viewpoint 65](#_Toc263679700)

[Hình 3-32 Các khái niệm sử dụng trong Actor Cooperation Viewpoint 66](#_Toc263679701)

[Hình 3-33 Ví dụ về Actor Cooperation Viewpoint 66](#_Toc263679702)

[Hình 3-34 Các khái niệm sử dụng trong Business Function Viewpoint 67](#_Toc263679703)

[Hình 3-35 Ví dụ về Business Function Viewpoint 68](#_Toc263679704)

[Hình 3-36 Các khái niệm sử dụng trong Product Viewpoint 69](#_Toc263679705)

[Hình 3-37 Ví dụ về Product Viewpoint 69](#_Toc263679706)

[Hình 3-38 Các khái niệm sử dụng trong Service Realisation Viewpoint 70](#_Toc263679707)

[Hình 3-39 Ví dụ về Service Realisation Viewpoint 70](#_Toc263679708)

[Hình 3-40 Các khái niệm sử dụng trong Business Process Collarboration Viewpoint 72](#_Toc263679709)

[Hình 3-41 Ví dụ về Business Process Collarboration Viewpoint 72](#_Toc263679710)

[Hình 3-42 Các khái niệm sử dụng trong Business Process Viewpoint 73](#_Toc263679711)

[Hình 3-43 Ví dụ về Business Process Viewpoint 73](#_Toc263679712)

[Hình 3-44 Các khái niệm sử dụng trong Information Structure Viewpoint 74](#_Toc263679713)

[Hình 3-45 Ví dụ về Information Structure Viewpoint 75](#_Toc263679714)

[Hình 3-46 Các khái niệm sử dụng trong Application Cooperation Viewpoint 76](#_Toc263679715)

[Hình 3-47 Ví dụ về Application Cooperation Viewpoint 77](#_Toc263679716)

[Hình 3-48 Các khái niệm sử dụng trong Application usage Viewpoint 78](#_Toc263679717)

[Hình 3-49 Ví dụ về Application usage Viewpoint 78](#_Toc263679718)

[Hình 3-50 Các khái niệm sử dụng trong *Application Behaviour Viewpoint* 79](#_Toc263679719)

[Hình 3-51 Ví dụ về *Application Behaviour Viewpoint* 80](#_Toc263679720)

[Hình 3-52 Các khái niệm sử dụng trong Application Structure Viewpoint 81](#_Toc263679721)

[Hình 3-53 Ví dụ về Application Structure Viewpoint 81](#_Toc263679722)

[Hình 3-54 Các khái niệm sử dụng trong Infrastructure Viewpoint 82](#_Toc263679723)

[Hình 3-55 Ví dụ về Infrastructure Viewpoint 83](#_Toc263679724)

[Hình 3-56 Ví dụ về Infrastructure Usage viewpoint 84](#_Toc263679725)

[Hình 3-57 Các khái niệm sử dụng trong Infrastructure Usage viewpoint 84](#_Toc263679726)

[Hình 3-58 Các khái niệm sử dụng trong Implement & Deloyment Viewpoint 85](#_Toc263679727)

[Hình 3-59 Ví dụ về Implement và Deloyment Viewpoint (Enterprise Architecture At Work) 86](#_Toc263679728)

[Hình 3-60 Cấu trúc Layer Viewpoint 87](#_Toc263679729)

[Hình 3-61 Ví dụ về Layer Viewpoint 87](#_Toc263679730)

[Hình 3-62 Bussiness Actor cooperation view 95](#_Toc263679731)

[Hình 3-63 Bussiness Organisation structure view 96](#_Toc263679732)

[Hình 3-64 Business function view 97](#_Toc263679733)

[Hình 3-65 Business product view 98](#_Toc263679734)

[Hình 3-66 Bussiness Service realisation view 99](#_Toc263679735)

[Hình 3-67 Business process cooperation view 99](#_Toc263679736)

[Hình 3-68 Bussiness process view 100](#_Toc263679737)

[Hình 3-69 Application cooperation view 101](#_Toc263679738)

[Hình 3-70 Application structure view 102](#_Toc263679739)

[Hình 3-71 Application behavior view 103](#_Toc263679740)

[Hình 3-72 Application usage view 104](#_Toc263679741)

[Hình 3-73 Technical Infrastructure View 105](#_Toc263679742)

[Hình 3-74 Layer view model 106](#_Toc263679743)

# TÓM TẮT

## Tên đề tài

“TÌM HIỂU ARCHIMATE VÀ XÂY DỰNG HỆ THỐNG QUẢN LÝ TRƯỜNG TRUNG HỌC PHỔ THÔNG”

## Nội dung nghiên cứu

Mục tiêu đề tài là nghiên cứu một ngôn ngữ kiến trúc hệ thống ở cấp độ Enterprise và áp dụng ngôn ngữ này vào thiết kế kiến trúc hệ thống quản lý trường trung học phổ thông cụ thể là trường THPT Trương Vĩnh Ký, TP HCM.

Do đó đề tài sẽ nghiên cứu những kiến thức cơ bản về kiến trúc Enterprise. Sau đó sẽ đi sâu vào ngôn ngữ kiến trúc hệ thống đó là ngôn ngữ Archimate. Và cuối cùng dùng Archimate để mô hình kiến trúc hệ thống quản lý trường trung học phổ thông.

## Hướng tiếp cận và giải quyết vấn đề

Nghiên cứu các tài liệu kỹ thuật liên quan.

Xem các ứng dụng liên quan đến nghiệp vụ quản lý trường học.

## Một số kết quả đạt được

Biết được kiến thúc cơ bản về kiến trúc Enterprise.

Nắm bắt phương pháp kiến trúc hệ thống theo ngôn ngữ Archimate.

Hiện thực mô hình kiến trúc bài toán quản lý trường trung học phổ thông.

# MỞ ĐẦU

* 1. **LÝ DO CHỌN ĐỀ TÀI**

Kiến trúc Enterprise là một hướng phát triển mới lâu dài và bền vững. Hiện tại trên thế giới, để trở thành một kiến trúc sư Enterprise là một việc không hề đơn giản, số lượng kiến trúc sư Enterprise đếm trên đầu ngón tay.

Một ngôn ngữ kiến trúc trực quan dễ hiểu giúp ta có cái nhìn tổng quan và sâu sắc kiến trúc của một hệ thống là vô cùng cần thiết. Giống như việc kiến trúc một căn nhà, các kiến trúc sư xây dựng cũng có một ngôn ngữ riêng để thiết kế. Trong lĩnh vực công nghệ thông tin cũng vậy, các kiến trúc sư cũng cần có một ngôn ngữ để kiến trúc hệ thống.

Ai cũng từng trải qua thời học sinh đầy mộng mơ và biết bao kỷ niệm. Mái trường thân quen ngày nào, sân trường, lớp học, thầy cô, bạn bè … vẫn còn động mãi trong tâm trí của chúng em. Chính mái trường phổ thông ấy đã chấp cánh cho chúng em đến với con đường tri thức rộng mở. Và để ngày hôm nay, chúng em được ngồi trong mái trường đại học, được làm luận văn tốt nghiệp. Vậy tại sao lại không chọn một luận văn nào đó giúp ích cho mái trường phổ thông thân yêu của chúng em.

Tất cả những điều trên chính là lý do chúng em chọn đề tài này.

* 1. **MỤC TIÊU ĐỀ TÀI**

Tìm hiểu kiến thức cơ bản về kiến trúc Enterprise.

Biết dùng ngôn ngữ Archimate trong kiến trúc hệ thống.

Xây dựng một ứng dụng quản lý trường trung học phổ thông sao cho ứng dụng có khả năng tùy biến cao, kiến trúc linh hoạt dễ thay đổi nếu như có yêu cầu mới.Ứng dụng có thể kết nối dữ liệu giữa các trường với nhau.

* 1. **PHẠM VI NGHIÊN CỨU**

Trong thời gian có hạn, đề tài của chúng em chỉ có thể thực hiện các hạng mục sau:

1. Các khái niệm cơ bản về Enterprise Architecture.
2. Tìm hiểu ngôn ngữ Archimate và ứng dụng để mô hình kiến trúc hệ thống quản lý trường trung học phổ thông.
3. Tìm hiểu kiến trúc SOA, web service.
4. Hiện thực các module quản lý giáo viên, quản lý học sinh, quản lý tiền thu học phí.

# TỔNG QUAN

* 1. **ĐẶT VẤN ĐỀ**

Kiến trúc Enterprise là một chủ đề đã được biết đến từ lâu. Tuy nhiên để hiểu và ứng dụng được nó vào thực tiễn sản xuất thì không phải là điều đơn giản.Các công ty, các tổ chức lớn , các tập đoàn công nghiệp… là các đối tượng quan tâm đến điều này.

Một hệ thống lớn muốn vận hành tốt và hiệu quả thì chúng ta phải hiểu và kiếm soát được nó. Để làm được điều đó cần có một kiến trúc tốt. Kiến trúc Enterprise sẽ mở ra một hướng tiếp cận mới giúp doanh nghiệp có cái nhìn tổng quát và sâu sắc hệ thống của mình. Tuy nhiên làm thế nào để có một kiến trúc Enterprise hoàn chỉnh thật sự là một vấn đề quá lớn và lâu dài không thể đề cập trong tài liệu này được.

Một vấn đề không kém phần quan trọng trong kiến trúc Enterprise là việc truyền thông giữa các bên liên quan đến hệ thống. Để làm được điều đó cần có một ngôn ngữ kiến trúc rõ ràng, mạnh mẽ và dễ hiểu. Nói đến đây chắc hẳn mọi người sẽ nghĩ ngay đến UML một ngôn ngữ rất quen thuộc và nổi tiếng. Tuy nhiên ở cấp độ một hệ thống Enterprise vô cùng lớn thì UML chưa đáp ứng được yêu cầu.

Trong phạm vi của đề tài sẽ trình bày kiến thức cơ bản về kiến trúc Enterprise và một ngôn ngữ kiến trúc hệ thống mới mẻ đó là ngôn ngữ Archimate.

# NỘI DUNG NGHIÊN CỨU

* 1. **Giới thiệu cơ bản về Enterprise Architecture**
     1. **Các thuật ngữ chuyên môn**

**Architecture**-trong một nghĩa rộng thì nó là sự tổng hợp giữa nghệ thuật và khoa học trong việc thiết kế một cấu trúc phức tạp mà các chức năng và sự phức tạp được kiểm soát.Theo nghĩa hẹp hơn thì nó là thành phần cơ bản của hệ thống và được thể hiện trong chính các thành phần của nó, các mối quan hệ giữa các thành phần đó với nhau và với môi trường xung quanh, các nguyên tắc đưa ra thiết kế và sự phát triển của hệ thống.

**Stakeholder** - Là cá nhân, nhóm người hay tổ chức có liên quan đến hệ thống, hoặc quan tâm đến hệ thống.

Hầu hết các stakeholder không quan tâm đến kiến trúc của hệ thống mà họ chỉ quan tâm đến lợi ích hệ thống mang lại cho họ. Người kiến trúc sư cần phải nhận ra những mong muốn của các stakeholder và giải thích cho họ hiểu rõ về hệ thống theo cách nhìn phù hợp với họ, cái đó người ta gọi là các viewpoint.

**Enterprise** – Là một tập hợp các tổ chức có cùng chung một mục đích. Hoặc một nhóm người được tổ chức để tạo ra sản phẩm, sử dụng công nghệ.

**Enterprise architecture** – Là một tập tất cả các nguyên tắc chặt chẽ, các phương thức và những mô hình mà được sử dụng trong thiết kế và hiện thực kiến trúc, quy trình nghiệp vụ, thông tin hệ thống, và cơ sở hạ tầng của một tổ chức enterprise.

Kiến trúc ở cấp độ là toàn bộ tổ chức thì thường được gọi là ‘enterprise architecture’. Ở đây chúng ta có thể hình dung mức độ của một tổ chức thật lớn cỡ như cả một tập đoàn, một tổ chức cấp quốc gia hoặc liên lục địa. Ví dụ như kiến trúc của hệ thống quản lý thuế của nước Việt Nam cũng được gọi là enterprise architecture.

**Driver** – Có nghĩa là động lực. Có những việc, những hoàn cảnh hay sự kiện nào đó xảy ra làm ảnh hưởng đến cách hoạt động của tổ chức doanh nghiệp theo một cách tích cực hoặc tiêu cực thì được gọi là ‘Driver ’. Có hai loại động lực đó là ‘động lực bên trong’ và ‘động lực bên ngoài’.

**Internal Drivers –** Động lực bên trong. Những việc, những hoàn cảnh hay sự kiện nào đó xảy ra bên trong doanh nghiệp và nói chung chúng nằm dưới sự kiểm soát của doanh nghiệp thì được gọi là internal drivers. Ví dụ như : máy móc, trang thiết bị của doanh nghiệp, năng lực kỹ thuật, văn hóa của doanh nghiệp, hệ thống quản lý, quản lý tài chính hay thậm chí là tinh thần làm việc của nhân viên…

**External Drivers –** Động lực bên ngoài. Những việc, những hoàn cảnh hay sự kiện nào đó xảy ra bên ngoài doanh nghiệp và nằm ngoài tầm kiểm soát của doanh nghiệp thì được gọi là external drivers. Ví dụ như : nền kinh tế thị trường, sự cạnh tranh giữa các doanh nghiệp…

External drivers có thể giết chết một tổ chức nếu như tổ chức đó không biết phản ứng một cách thích đáng đối với các tác động từ external drivers. Câu hỏi được đặt ra là làm thế nào một tổ chức có thể biết được những thay đổi gì đang xảy ra để có thể thích ứng một cách tích cực.

* + 1. **Tại sao phải có kiến trúc enterprise**

Kiến trúc lưu giữ các yếu tố cần thiết của tổ chức, ứng dụng công nghệ thông tin vào hoạt động và sự phát triển của nó. Ý tưởng là yếu tố cần thiết và ổn định hơn là các giải pháp cụ thể được tìm thấy để giải quyết các vấn đề trong tầm tay. Do đó kiến trúc rất hữu ích trong việc bảo vệ các yếu tố cần thiết của tổ chức, trong khi vẫn cho phép sự linh hoạt và thích ứng tối đa. Nếu không có kiến trúc tốt thì sẽ là khó khăn để đạt được thành công cho tổ chức.

Đặc điểm quan trọng nhất của kiến trúc enterprise là nó cung cấp một cái nhìn tổng thể của tổ chức. Một kiến trúc enterprise tốt cung cấp cái nhìn sâu sắc, thấu đáo cần thiết trong việc cân bằng các yêu cầu và biến chuyển dễ dàng từ tổ chức chiến lược đến hoạt động hằng ngày.

Ngoài việc cung cấp cái nhìn tổng thể thì kiến trúc enterprise còn được sử dụng để ước lượng, đánh giá quá trình chuyển đổi từ hiện tại đến tương lai. Nó cung cấp một phương pháp để đánh giá tác động của những thay đổi kiến trúc đến chất lượng sản phẩm và cả khía cạnh số lượng, ví dụ như hiệu năng sản xuất hay vấn đề chi phí.

Mục đích cuối cùng của kiến trúc Enterprise là giải quyết bài toán áp dụng IT vào trong các quy trình nghiệp vụ của tổ chức nhằm nâng cao hiệu quả công việc.

* + 1. **Kiến trúc sư enterprise – EA**

CIO là chức danh của một người lãnh đạo CNTT ở các công ty đa quốc gia hay tập đoàn, công ty… Nhưng không phải CIO nào cũng có thể xây dựng hệ thống thông tin vừa linh hoạt, vừa là nền tảng cho DN sử dụng trong hiện tại và phát triển tốt trong tương lai. Gần đây, trên thị trường lao động, có một chức danh khác chỉ người đóng vai “đầu tàu” CNTT của các tổ chức, DN lớn, vừa và nhỏ, một cái tên gần gũi nhưng cũng “rất oai”: Enterprise Architect (EA – kiến trúc sư doanh nghiệp).

Simon Guest, giám đốc bộ phận Architecture Strategy của Microsoft giải thích về EA: “Vai trò của EA là đảm bảo định hướng của DN và CNTT phải đi cùng hướng (alignment). EA cố gắng để tối đa hóa lợi ích đầu tư cho CNTT bằng cách hoặc ưu tiên chi tiêu cho lợi ích kinh doanh hoặc làm tác động của CNTT lên dịch vụ, tài nguyên, dự án và quy trình của DN là cao nhất”.

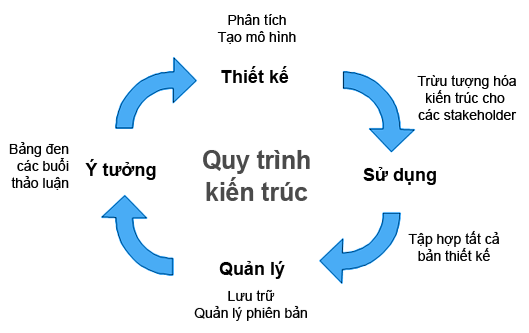
EA ví như cầu nối giữa 3 thành phần: lãnh đạo, phát triển và điều hành DN để đảm bảo rằng cả 3 hiểu nhau, trả lời câu hỏi “mục tiêu DN đặt ra khả thi hay không?” đồng thời kiểm soát được các vấn đề phát sinh. Tầm nhìn của EA bao quát như người lãnh đạo DN, nhiệm vụ giải quyết mối liên hệ con người và công nghệ nhằm tạo ra được sản phẩm có đẳng cấp quốc tế, những tầm nhìn dài hơi. Và, đôi khi EA còn được gọi là Strategic Architect – kiến trúc sư chiến lược.

* + 1. **Quy trình kiến trúc hệ thống**

Đầu tiên là các những ý tưởng được ghi nhận lại, có thể là viết trên bảng đen, hay qua slide powerpoint … người kiến trúc sư sẽ phân tích và thiết kế các ý tưởng đó thành các mô hình kiến trúc.

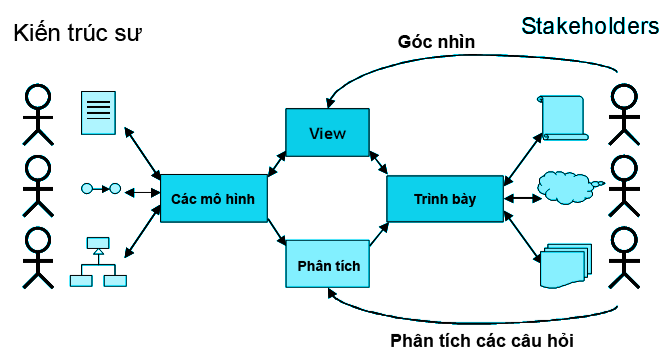
Kế tiếp kiến trúc sư sẽ đưa ra các bảng vẻ, các tài liệu, các mô hình cho các stakeholder. Tùy vào từng stakeholder khác nhau mà có tài liệu phù hợp với stakeholder đó.

Cuối cùng, sau khi đã thống nhất đồng ý với các thiết kế thì kiến trúc sư sẽ tập hợp tất cả các bảng vẽ, các tài liệu thiết kế lại thành một tập phiên bản và quản lý từng phiên bản thiết kế. Và cứ như vậy phiên bản của các bản thiết kế cứ phát triển dần dần.



1. Vòng đời của quy trình kiến trúc hệ thống
   * 1. **Sự truyền thông trong quá trình kiến trúc Enterprise**

Đối với quá trình kiến trúc và xây dựng một hệ thống thông tin thì sự truyền thông là không thể thiếu được. Thành công hay thất bại của dự án đều bị ảnh hưởng rất lớn từ việc truyền thông giữa các thành viên liên quan đến dự án có hiệu quả hay không. Quá trình truyền thông này không chỉ diễn ra xuyên suốt từ lúc bắt đầu cho đến lúc hoàn thành dự án, mà nó còn luôn được diễn ra trong quá trình duy trì và phát triển sau đó. Vì vậy truyền thông hiệu quả là một trong những yêu cầu thiết yếu đối với kiến trúc enterprise.



1. Sự truyền thông giữa Enterprise Architects với các Stakeholders

Trong ngữ cảnh kiến trúc, thì một bản mô tả chi tiết về kiến trúc (*describing architecture hay còn gọi là architecture description)* là phương tiện quan trọng hỗ trợ chúng ta trong việc truyền thông. Một ví dụ về đặc tả kiến trúc, nó có thể bao gồm tất cả các vấn đề như: mô tả hệ thống và sự phát triển của nó, phân tích kiến trúc bên trong hệ thống, kế hoạch kinh doanh mới cho việc chuyển đổi từ kiến trúc cũ sang kiến trúc mới, sự truyền thông giữa những tổ chức trong quá trình phát triển, sản xuất, duy trì hệ thống, sự thông tin giữa bên yêu cầu và bên phát triển… Đặc tả kiến trúc được sử dụng để thông tin về kiến trúc của hệ thống dự kiến hoặc một hệ thống đã sẵn có. Nó có thể là một phần của một enterprise, một tổ chức, một quy trình, một hệ thống thông tin, hoặc cơ sở hạ tầng kỹ thuật.

* Cộng đồng phát triển hệ thống - System Development Community

Khi nói về sự truyền thông, một nhiệm vụ quan trọng là nhận diện được những actor, những người sẽ đóng một vai trò nào đó trong việc truyền thông trong suốt quá trình phát triển hệ thống của chúng ta, ví dụ như là các chuyên viên, khách hàng, kiến trúc sư, kỹ sư, nhà phân tích kinh doanh…Tuy nhiên, không chỉ những actor đóng vai trò quan trọng trong việc truyền thông, mà bên cạnh đó còn có những lớp đối tượng quan trọng khác như document, model, form…những đối tượng này miêu tả mẫu và những phần của kiến thức liên quan đến hệ thống đang được phát triển. Tất cả những đối tượng trên được xem là “system development community”

***System development community***: Là môt nhóm những đối tượng được yêu cầu trong quá trình phát triển hệ thống như là actor, presentation.

Những actor trong “system development community” sẽ có những mối quan tâm nhất định đối với những vấn đề liên quan đến hệ thống, những mối quan tâm này tường được gọi là *concern* của stakeholder.

***Concern***: Là mối quan tâm của stakeholder về những vấn đề liên quan đến đặc tả kiến trúc của hệ thống, kết quả từ những mục đích của stakeholder, và vai trò hiện tại và tương lai của hệ thống trong mối quan hệ với những mục đích này.

* Kiến thức phát sinh - System Development Knowledge

Trong suốt quá trình phát triển hệ thống, những kiến thức mới sẽ được phát sinh ra thông qua quá trình trao đổi thông tin giữa các thành viên trong cộng đồng phát triển hệ thống, được gọi là System Development Knowledge.

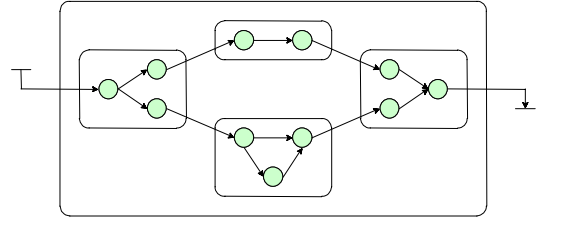
Sự trao đổi thông tin thông gắn liền với các chủ đề khác nhau nhau. Thông qua các chủ đề mà các kiến thức phát sinh được làm rõ ràng hơn.

* Trạng thái của kiến thức - Transformations of Knowledge

Mỗi kiến thức phát sinh đều trải qua ba giai đoạn:

* ***Aware:***giai đoạn giới thiệu kiến thức mới mà một actor đúc kết và trình bày tới các actor khác.
* ***Agreed:*** Khi kiến thức được chia sẽ, một actor có thể thể hiện suy nghĩ của mình về kiến thức đó, và quyết định có đồng ý hay không với kiến thức được chia sẽ này.
* ***Committed:***Khi những actor đã đồng ý với kiến thức này thì trạng thái đạt được của kiến thức này là “*được ủy thác*”.
* Chiến lược tổ chức hội thảo

Như đã thảo luận trước, sự biến chuyển của kiến thức là kết quả của những cuộc hội thảo, những cuộc hội thảo này là một chuỗi từ những thảo luận nhỏ chỉ gồm một vài actor cho đến những hội thảo lớn với toàn bộ workgroup.



1. Ví dụ về một chuỗi các hội thảo

* Knowledge Goals

Như đã trình bày ở những phần trước,trong ngữ cảnh truyền thông, kiến thức gồm ba trang thái lớn: awareness, agreement, commitment. Dựa trên sự điều này, kiến thứcgoal chính trạng thái nào đó(trong ba trạng thái) của kiến thức mà bạn muốn đạt được từ cuộc hội thảo.

* Xác định chiến lược hội thảo

Mỗi cuộc hội thảo điều được diễn ra để đạt được một *knowledge goal* nào đó, có nghĩa là một trang thái nào đó của kiến thức mà một cuộc hội thảo nhắm đến, và để đạt được những mục đích này, cuộc hội thảo sẽ phải theo một chiến lược hội thảo nào đó.

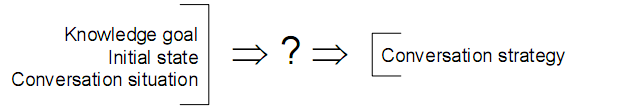
Một cuộc hội thảo thường phải đặt trong một ngữ cảnh nào đó, và điều này có ảnh hưởng nhất định đến cuộc hội thảo. Chúng ta sẽ giới thiệu một số đặc điểm về những ngữ cảnh như vậy:

* **Availability of resources**: Liên quan đến những tài nguyên(cơ sở vật chất hạ tầng, tài nguyên về con người, tài chính…) sẳn có để sử dụng trong một cuộc hội thảo
* **Complexity**: Liên quan đến tài nguyên cần thiết, kiến thức cần được trao đổi…Yếu tố này có thể tác động đến chiến lược hội thảo. Ví dụ của sự phức tạp ở đây như: về số lượng những actor, những actor khác nhau về kiến thức, trình độ, về sự phức tạp của kiến thức cần thảo luận, sự phức tạp của công nghệ cần sử dụng…
* **Uncertainty**: Để xác định được một chiến lược hội thảo, trước tiên chúng ta cần phải xác định được hoàn cảnh hiện tại, đưa ra những nhận định về *knowledge goal*, trạng thái của kiến thức, xác định tài nguyên sẵn có…Trong quá trình thực thi cuộc hội thảo, những sự xác định này có thể sai, hoặc không chắc chắn.

Những đặc điểm trên rất quan trọng trong việc xác định một chiến lược hội thảo phù hợp. Thêm vào đó, một chiến lược hội thảo luôn bao gồm ít nhất những yếu tố sau:

* **Execution plan**: Một cuộc hội thảo có thể gồm nhiều cuộc thảo luận nhỏ hơn, Mỗi cuộc thảo luận nhỏ này thường tập trung vào một mục tiêu nhỏ, nhưng tất cả chúng đều hướng về mục đích chung của cuộc hội thảo. Execute plan của một cuộc hội thảo bao gồm một tập hợp những cuộc hội thảo nhỏ với sự thực thi theo một kế hoạch định trước .
* **Description languages**: Ngôn ngữ được sử dụng trong cuộc hội thảo.
* **Media**: Loại phương tiện được sử dụng trong suốt một cuộc hội thảo.
* **Cognitive mode**: Yêu cầu cách mà những kiến thức được tập hợp hoặc xử lý bởi yêu cầu của actor.
* **Social mode**: Cách mà actor hiện thực quy trình phát triển hệ thống kết hợp với actor từ lĩnh vực kinh doanh.
* **Communication mode**: Phân biệt một số mẫu được sử dụng trong quá trình thông tin.

Tổng kết lại, chúng ta thấy rằng từ *knowledge goal, initial state, conversation consituation,* ta có thể xác định được *conversation stategy*:



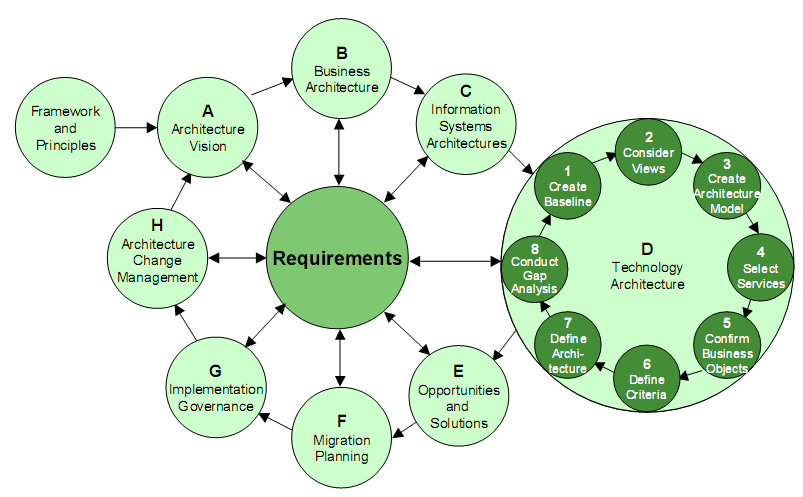
Với việc đưa ra những cuộc chiến lược hội thảophù hợp với hoàn cạnh hiện tại, chúng ta sẽ có được những sự truyền thông hiệu quả, Điều này góp phần quyết định thành công của dự án.

* + 1. **Các phương pháp kiến trúc**

**RUP**– viết tắc của cụm từ Rational Unified Process. Nó định nghĩa một quy trình lặp đi lặp lại. Trái ngược với quy trình thác nước cổ điển, nó hiện thực phần mềm bằng cách thêm các chức năng mới vào kiến trúc mỗi khi ra phiên bản mới.

**UMM** - UN/CEFACT Modelling Methodology. Nó là một quy trình hoạt động gia tăng và là phương pháp xây dựng mô hình thông tin. Business Collaboration Framework (BCF) là một framework đang được phát triển và sẽ là chuyên ngành của UMM nhằm xác định sự trao đổi thông tin bên ngoài của tổ chức và cá hoạt động nghiệp vụ cơ bản của nó.

**ADM** - The TOGAF Architecture Development Method. Được phát triển bởi Open Group, nó cung cấp chi tiết và mô tả giai đoạn phát triển một kiến trúc công nghệ thông tin.



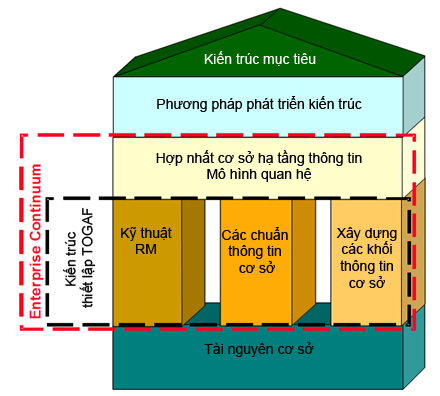
1. Vòng đời phát triển kiến trúc của phương pháp TOGAF
   * 1. **Các framework hỗ trợ**

**Zachman Framework** – framework được John Zachman giới thiệu đầu tiên năm 1987 và được coi là một enterprise architecture framework nổi tiếng nhất.



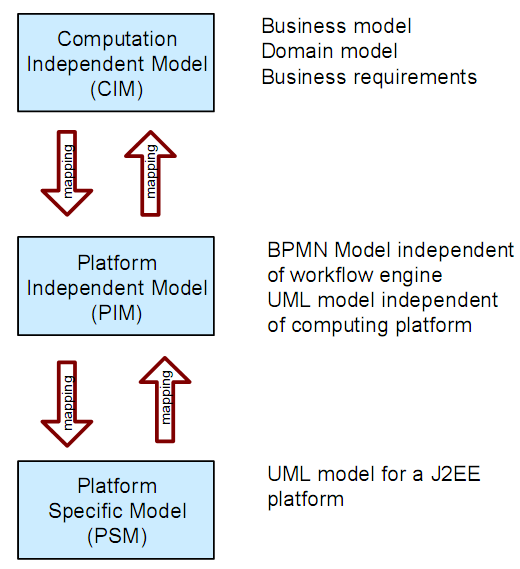
1. Zachman Framework (Zachman 1987)

**The Open Group Architecture Framework (TOGAF)**



1. TOGAF (based on The Open Group 2002).

**Model-Driven Architecture (MDA)**



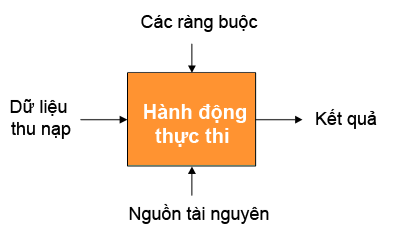
1. MDA framework.
   * 1. **Các ngôn ngữ kiến trúc hệ thống**

* IDEF

IDEF - là tên gọi chung của một họ các ngôn ngữ được sử dụng để phân tích và hiện thực mô hình kiến trúc enterprise. IDEF là một nhóm các phương pháp có nguồn gốc từ quân đội. Ban đầu, chúng được phát triển bởi lực lượng không quân Mỹ trong chương trình *Integrated Computer Aided Manufacturing (ICAM)*.

Hiện tại, có 16 phương pháp IDEF. Trong những phương pháp này, IDEF0, IDEF3 và IDEF1X ( “hạt nhân”) thì được sử dụng phổ biến nhất.

***Functional modelling, IDEF0***- Mô hình chức năng: ý định của IDEF0 là mô hình hóa những yếu tố đang kiểm soát việc thực hiện một chức năng A nào đó, những actor hiện thực chức năng A này, những đối tượng hay dữ liệu được thu nạp và được tạo ra bởi chức năng này và những mối quan hệ giữa những chức năng nghiệp vụ với chức năng A.



1. IDEF0 representation.

***Process modelling, IDEF3***- Mô hình tiến trình: IDEF3 lưu lại tiến trình công việc của mộ quy trình nghiệp vụ thông qua biểu đồ tiến trình *process flow diagrams*. Những tiến trình này biểu thị một trình tự các tiến trình được thực hiện của tổ chức, những quyết định logic, mô tả các kịch bản khác nhau để thực hiện các chức năng nghiệp vụ giống nhau, và có thể phân tích và cải thiện quy trình công việc.

***Data modelling, IDEF1X***– Mô hình dữ liệu : IDEF1X được sử dụng để tạo ra những mô hình dữ liệu hợp logic và những mô hình dữ liệu vật lý.

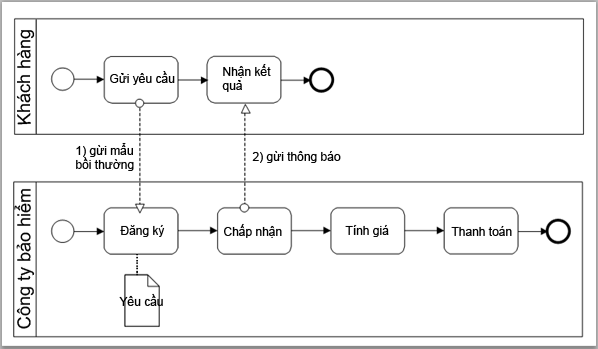
Hệ thống IDEF cung cấp sự hỗ trợ cho việc mô hình hóa một vài view trong kiến trúc. Tuy nhiên, không có sự liên hệ giữa những mô hình này. Sự thật rằng chúng bị tách ra gây cản trở cho việc hình dung các yếu tố có liên quan với nhau trong hệ thống kiến trúc. Điều này cũng có nghĩa rằng một sự chuyển đổi giữa các view là không thể.

Tuy nhiên IDEF được sử dụng rộng rãi trong lĩnh vực công nghiệp vì nó đáp ứng nhu cầu của người sử dụng trong một giới hạn chấp nhận được.

* Business Process Modelling Notation (BPMN)

BPMN là một trong những tiêu chuẩn đang được phát triển bởi tổ chức BPMI - Business Process Management Initiative.

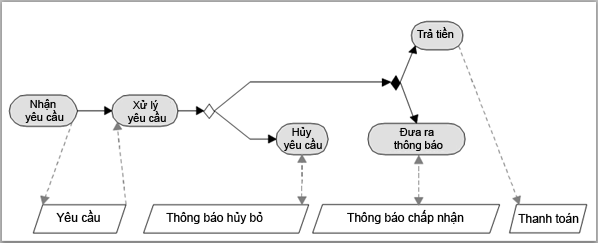
BPMN bị giới hạn trong việc tạo mô hình. Tầng ứng dụng và cơ sở hạ tầng không được thể hiện bởi BPMN. Mục đích chính của nó là cung cấp một ký hiệu thống nhất cho việc mô hình quy trình nghiệp vụ (hình 3-8).



1. Ví dụ mô hình được thể hiện bởi BPMN

* Testbed

Testbed là ngôn ngữ mô hình quy trình nghiệp vụ và phương pháp đầu tiên được phát triển bởi Telematica Instituut cùng với nhiều công ty được liên kết với nhau. Testbed hiện giờ được sử dụng bởi nhiều công ty và Chính phủ Hà Lan.

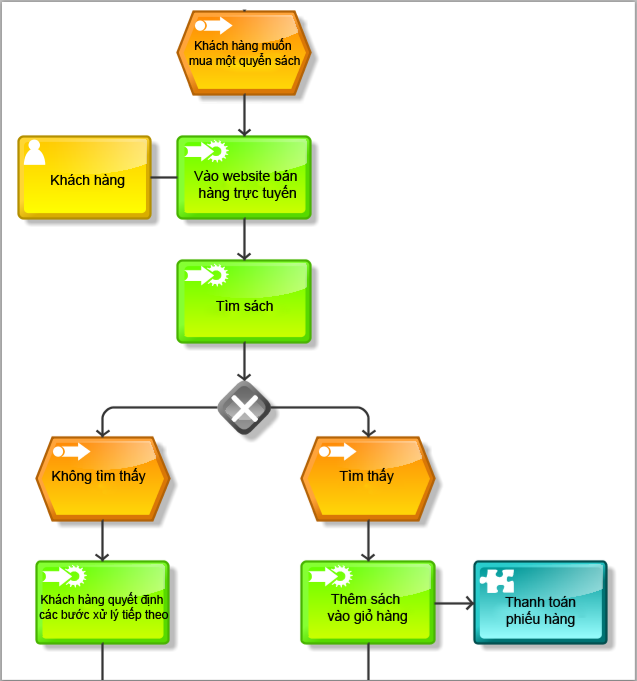


1. Mô hình quy trình nghiệp vụ trong Testbed.

* ARIS (‘Architecture of Integrated Information Systems’, Scheer 1994)

ARIS là một phương pháp nổi tiếng để mô hình kiến trúc Enterprise. ARIS được nhắm đến để phục vụ cho những mục đích khác nhau: cung cấp tài liệu của các loại quy trình nghiệp vụ hiện có, lên kế hoạch cho việc phân tích và thiết kế những quy trình nghiệp vụ và hổ trợ thiết kế những hệ thống thông tin.

Để mô hình quy trình nghiệp vụ trong tổ chức, ARIS cung cấp một ngôn ngữ mô hình.Hình bên dưới cho ta một ví dụ mô hình tiến trình công việc được tạo ra bởi ‘ARIS Toolset’ một công cụ giúp mô hình kiến trúc theongôn ngữ ARIS.



1. Quy trình thanh toán hóa đơn thể hiện bởi ARIS.

ARIS cung cấp các loại mô hình sau:

* Mô hình dữ liệu (data view)
* Mô hình luồn công việc (control flows)
* Mô hình quy trình, chức năng nghiệp vụ (process / funtion view)
* Mô hình tổ chức (organisation view)
* Unified Modeling Language (UML)

UML - Ngôn ngữ mô hình hóa thống nhất : là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

Cách tiếp cận theo mô hình của UML giúp ích rất nhiều cho những người thiết kế và thực hiện hệ thống thông tin cũng như những người sử dụng nó; tạo nên một cái nhìn bao quát và đầy đủ về hệ thống thông tin dự định xây dựng. Cách nhìn bao quát này giúp nắm bắt trọn vẹn các yêu cầu của người dùng; phục vụ từ giai đoạn phân tích đến việc thiết kế, thẩm định và kiểm tra sản phẩm ứng dụng công nghệ thông tin.

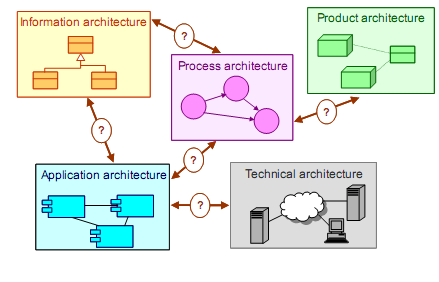
UML sử dụng một hệ thống ký hiệu thống nhất biểu diễn các *Phần tử mô hình* (*model elements*). Tập hợp các phần tử mô hình tạo thành các *Sơ đồ UML* (*UML diagrams*). Có các loại sơ đồ UML chủ yếu sau:

* Sơ đồ lớp (Class Diagram)
* Sơ đồ đối tượng (Object Diagram)
* Sơ đồ tình huống sử dụng (Use Cases Diagram)
* Sơ đồ trình tự (Sequence Diagram)
* Sơ đồ cộng tác (Collaboration Diagram hay là Composite Structure Diagram)
* Sơ đồ trạng thái (State Machine Diagram)
* Sơ đồ thành phần (Component Diagram)
* Sơ đồ hoạt động (Activity Diagram)
* Sơ đồ triển khai (Deployment Diagram)
* Sơ đồ gói (Package Diagram)
* Sơ đồ liên lạc (Communication Diagram)
* Sơ đồ tương tác (Interaction Overview Diagram - UML 2.0)
* Sơ đồ phối hợp thời gian (Timing Diagram - UML 2.0)



1. Ví dụ một mô hình UML
   1. **Archimate language**
      1. **ArchiMate là gì**

ArchiMate là một ngôn ngữ dùng để miêu tả mô hình kiến trúc enterprise,hỗ trợ kiến trúc sư trong việc mô tả, phân tích, hình dung, và truyền đạt các khái niệm kiến trúc, mối quan hệ, và những tác động bên trong và giữa các miền enterprise một cách rõ ràng và được chuẩn hóa.



1. Các miền trong ArchiMate

ArchiMate là một ngôn ngữ chung để mô tả, phân tích và mô hình các quy trình nghiệp vụ, cơ cấu tổ chức, dòng chảy của thông tin, hệ thống CNTT, cơ sở hạ tầng kỹ thuật.Điều này giúp các bên liên quan thiết kế, đánh giá, và truyền đạt thông tin cần trao đổi một cách dễ dàng hơn.

**ArchiMate** là một ngôn ngữ có khả năng mở rộng với nhiều khía cạnh :

* Nền tảng kiến trúc của nó đơn giản nhưng toàn diện, đủ để cung cấp một cơ cấu tốt cho các miền kiến trúc, các tầng, và các khía cạnh khác nhau.
* Ngôn ngữ này kết hợp những ý tưởng hiện đại của mô hình " định hướng dịch vụ ",là động lực để thúc đẩy các nguyên tắc tổ chức mới về dịch vụ (kinh doanh, ứng dụng, và cơ sở hạ tầng) cho các tổ chức, với những hiệu quả sâu rộng cho kiến trúc enterprise.
* Các ký hiệu ArchiMate mô hình trực quan hơn mô hình hiện thời của UML 2.0. Ngôn ngữ này đủ ý nghĩa để giúp mô hình của tất cả các tầng (nghiệp vụ, ứng dụng, và cơ sở hạ tầng) và tất cả các khía cạnh (cấu trúc, ứng xử và thông tin) của một tổ chức một cách thích hợp.
  + 1. **Tại sao dùng ArchiMate**

Ngày nay, các tổ chức cần phải đáp ứng nhanh chóng những yêu cầu thay đổi của khách hàng. Nhu cầu này ảnh hưởng đến toàn bộ hoạt động của một tổ chức, từ cơ cấu tổ chức đến cơ sở hạ tầng. Làm thế nào bạn kiểm soát các tác động của những thay đổi này? Kiến trúc có thể là câu trả lời.Tuy nhiên, việc mô hình một kiến trúc sao cho đúng đắn, hiệu quả và thống nhất thực sự là một thử thách không nhỏ, chẵng hạn như kiến trúc sư cần những cách thể hiện kiến trúc một cách rõ ràng để có thể liên lạc với các bên liên quan khác, chẳng hạn như bên phát triển hệ thống,người dùng cuối, và nhà quản lý...Thật không may, trên thực tế các kiến trúc sư thuộc các lĩnh vực khác nhau, thậm chí trong cùng một tổ chức, thường sử dụng riêng các kỹ thuật mô tả và quy ước riêng của họ. Đến nay, không có tiêu chuẩn ngôn ngữ để mô tả kiến trúc enterprise một cách chính xác. Chúng thường được mô tả hoặc trong hình thức thiếu một ý nghĩa rõ ràng, hoặc trong ngôn ngữ thiết kế chi tiết (như UML) gây khó hiểu cho những người không chuyên. Điều này thường xuyên dẫn đến hiểu lầm làm cản trở sự hợp tác của kiến trúc sư và các bên liên quan khác.

Ngôn ngữ mô hình kiến trúc ArchiMate là câu trả cho những khó khăn trên.Ngôn ngữ ArchiMate phát triển một phương pháp tiếp cận thích hợp để mô tả kiến trúc, hình dung ra các miền khác nhau của một enterprise và quan hệ giữa chúng.

ArchiMatethể hiện một cách thống nhất về mô hình kiến trúc enterprise, tích hợp các miền khác nhau và mô tả chúng một cách rõ ràng,dễ hiểu. Bằng cách phát triển một ngôn ngữ kiến trúc và kỹ thuật hình ảnh trực quan, ArchiMate sẽ cung cấp cho các kiến trúc sư các công cụ hỗ trợ và cải thiện quy trình kiến trúc hiện có và tiêu chuẩn mới sẽ được sử dụng hoặc tích hợp bất cứ khi nào có thể.

* + 1. **Những lợi ích của ArchiMate**

ArchiMate là một ngôn ngữ tiêu chuẩn quốc tế được cung cấp bởi Tập đoàn Open, giải phóng bạn thoát khỏi sự ràng buộc của các công cụ được chỉ định và các framework.Có hoạt động hỗ trợ của diễn đàn ArchiMate của Tập đoàn Open.

ArchiMate là những khái niệm và mô hình đã được biết đến,cung cấp sự chính xác. Nó giúp bạn nhận thức được các 'hình ảnh mơ hồ" về kiến trúc.

ArchiMate là một ngôn ngữ dễ hiểu và dễ áp dụng. Nó chứa chỉ đủ cho các khái niệm mô hình kiến trúc enterprise mà không bao gồm những thứ cồng kềnh không cần thiết khác. Cấu trúc đồng nhất của nó làm cho nó dễ dàng để học và áp dụng.

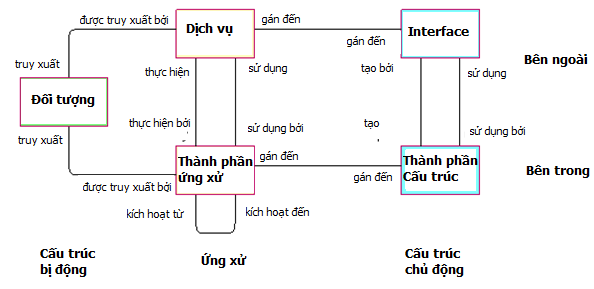
Archimate có liên kết rõ ràng đối với phương pháp tiếp cận hiện tại, các khu vực kiến trúc cụ thể như phần mềm hay quy trình kinh doanh. Một số khái niệm trong ArchiMate đã cố tình vay mượn từ các ngôn ngữ khác như UML hoặc BPMN, để cung cấp cầu nối giữa chúng dễ dàng.

Nó dễ dàng kết hợp với phương pháp hiện có như TOGAF.

Nó đã được kiểm nghiệm bởi các tổ chức khác nhau và được hỗ trợ bởi nhiều nhà tư vấn và các công cụ phần mềm.

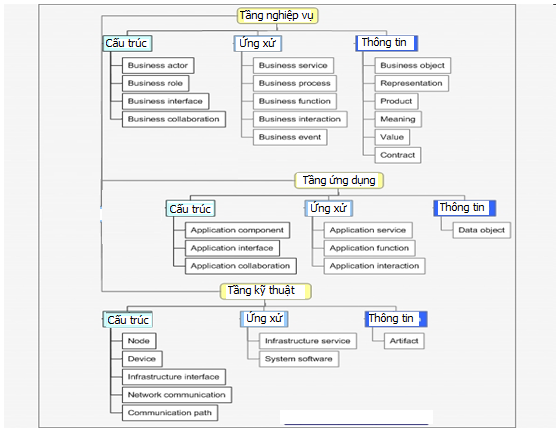
* + 1. **Các khái niệm chính trong ArchiMate**

Archimate gồm có ba tầng chính: *tầng nghiệp vụ, tầng ứng dụng, tầng vật lý*, mỗi tầng đều chứa những thành phần thuộc về các *khía cạnh cấu trúc(chủ động, bị động)*,*khía cạnh ứng xử*.



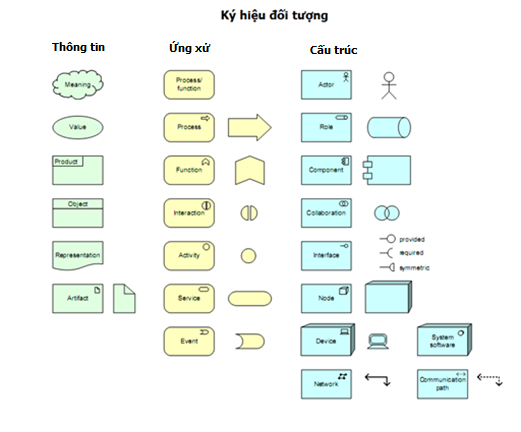
1. Cáckhái niệm chính trong ArchiMate

Hình bên dưới là lược đồ tổng quát về các khái niệm thuộc về các tầng và các khía cạnh trong ngôn ngữ ArchiMate:



1. Các khái niệm trong ArchiMate
   * 1. **Những ký hiệu của ArchiMate**

* Các ký hiệu cho những khái niệm trong ngôn ngữ

****

1. Những ký hiệu đối tượng

* Ký hiệu các mối quan hệ

Chúng ta sẽ được giới thiệu những mối quan hệ được dùng giữa những khái niệm trong ngôn ngữ Archimate. Những mối quan hệ này được phân làm ba loại:

+Structural Relationship-Mối quan hệ thuộc về cấu trúc

Được dùng để mô hình mối quan hệ giữa những khái niệm thuộc về cấu trúc với nhau hoặc giữa khái niệm thuộc về cấu trúc với những khái niệm khác loại.

|  |  |  |
| --- | --- | --- |
| Structural Relationship | | Ký hiệu |
| Association | Mô hình mối quan hệ giữa hai đối tượng, đối tượng này không bao gồm đối tượng kia |  |
| Access | Mô hình mối quan hệ truy xuất của những đối tượng thuộc về ứng xử đối với những business objecthoặc data object |  |
| Used by | Mô hình mối quan hệ sử dụng bởi ví dụ business service được sử dụng business process, hay business interface được sử dụng bởi business role |  |
| Realization | Mô hình mối quan hệ liên kết giữa một đối tượng luận lý với một đối tượng cụ thể hiện thực nó. |  |
| Assignment | Mối quan hệ này thể hiện liên kết giữa một đối tượng thuộc về ứng xử với một đối tượng chủ động thuộc về cấu trúc |  |
| Aggregation | Thể hiện mối quan hệ một đối tượng chứa một hay nhiều đối tượng khác |  |
| Composistion | Thể hiện mối quan hệ một đối tượng bao gồm một số đối tượng khác |  |

+Dynamic Relationsip-Mối quan hệ động

Được sử dụng để mô hình những mối quan hệ phụ thuộc giữa những khái niệm thuộc về ứng xử.

|  |  |  |
| --- | --- | --- |
| Dynamic Relationship | | Ký hiệu |
| Flow | Thể hiện sự trao đổi thông tin hay value giữa những đối tượng |  |
| Triggering | Thể hiện mối quan hệ về thời gian, hay mối quan hệ nguyên nhân kết quả giữa các đối tượng thuộc về ứng xử |  |

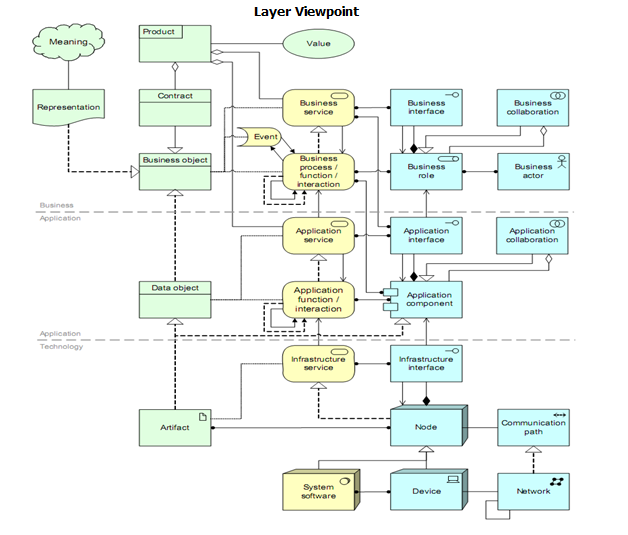
+Những mối quan hệ khác

|  |  |  |
| --- | --- | --- |
|  | | Ký hiệu |
| Grouping | Nhóm những đối tượng cùng loại hay khác loại dựa vào một đặc tính chung nào đó |  |
| Junction | Dùng để kết nối những mối quan hệ giống nhau |  |
| Specialization | Thể hiện mối liên kết mà một đối tượng chuyên môn hoá hay mở rộng đối tượng khác |  |

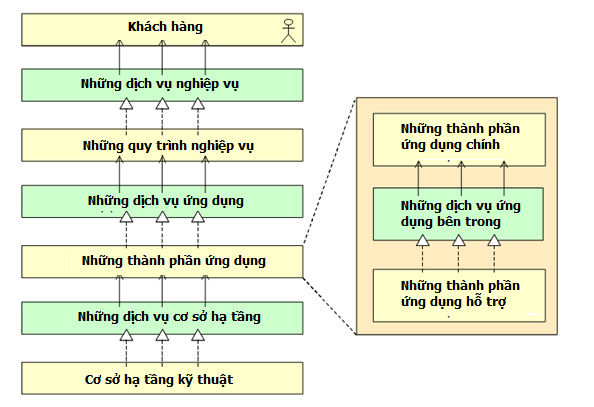
* + 1. **Kiến trúc tổng quát ngôn ngữ Archimate**

Trong phần này chúng ta sẽ đi tìm hiểu chi tiết về các tầng và các khía cạnh trong ngôn ngữ ArchiMate. Ngôn ngữ Archimate định nghĩa ba tầng chính :

* Tầng **Nghiệp vụ** : cung cấp sản phẩm và dịch vụ cho những khách hàng bên ngoài được nhận ra trong tổ chức của quy trình nghiệp vụ được thực hiện bởi các business actor.
* Tầng **Ứng dụng** : hỗ trợ các lớp Business với các dịch vụ ứng dụng được nhận ra bởi (phần mềm) ứng dụng.
* Tầng **Kỹ thuật** cung cấp dịch vụ cơ sở hạ tầng (ví dụ, xử lý, lưu trữ và các dịch vụ truyền thông) cần thiết để chạy các ứng dụng, thực hiện bởi máy tính và phần cứng và phần mềm hệ thống thông tin liên lạc.



1. Tổng quan về các khái niệm ArchiMate và mối quan hệ chính.



1. Mô hình phân tầng ArchiMate

* Bài toán mô hình ví dụ ArchiSurance

Để minh họa việc sử dụng ngôn ngữ mô hình hóa của chúng ta, chúng ta giới thiệu ví dụ về bài toánxây dựng hệ thống thông tin cho công ty bảo hiểm ArchiSurance. ArchiSurance cung cấp bảo hiểm nhà cửa và du lịch nhưng gần đây kết hợp với những công ty bảo hiểm khác, như PRO-FIT (bảo hiểm xe hơi) và Legally Yours (bảo hiểm viện trợ hợp pháp). Bằng cách sắp xếp hoạt động của mình và loại bỏ trùng lắp, sự đồng nhất giữa các công ty được mong đợi từ sự kết hợp.

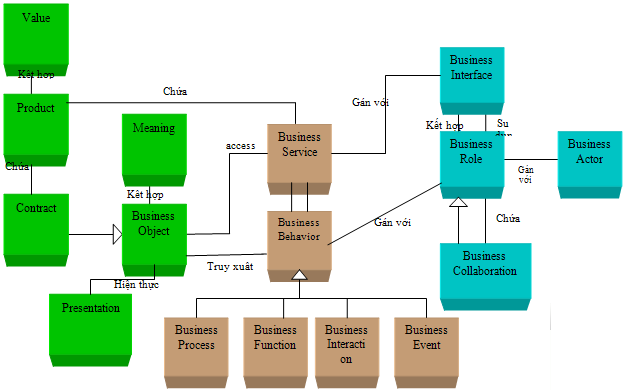
Việc quản lý ArchiSurance hiện giờ đang đối mặt với những phức tạp của việc tích hợp ba công ty, và đã quyết định đi một cách tiếp cận xây dựng kiến trúc doanh nghiệp để tạo ra cái nhìn sâu hơn phức tạp này.

Để cung cấp một cái nhìn tổng quan cấp cao của các hoạt động chính của ArchiSurance, công ty được mô tả về chức năng nghiệp vụ chính của nó:

* Duy trì quan hệ khách hàng và quan hệ môi giới: các nghiệp vụ có trách nhiệm liên lạc của ArchiSurance với khách hàng và các trung gian bán sản phẩm của mình. Chức năng này xử lý các câu hỏi của khách hàng, các yêu cầu được gửi đến, và thực hiện tiếp thị và bán hàng.
* Ký kết: chức năng này giúp cho các văn phòng xử lý,ký kết các hợp đồng. Nó phân tích rủi ro và đảm bảo về mặt pháp lý và chính xác các hợp đồng tài chính.
* Xử lý yêu cầu: chức năng này có trách nhiệm xử lý khiếu nại bảo hiểm.
* Xử lý tài chính: chức năng này thực hiện việc thu phí bảo hiểm thường xuyên, theo các chính sách bảo hiểm với khách hàng bằng cách ký kết, và xử lý việc thanh toán bồi thường bảo hiểm.
* Quản lý tài sản: chức năng này quản lý các tài sản tài chính của ArchiSurance, ví dụ, bằng cách đầu tư vào cổ phiếu và trái phiếu.

Ví dụ Archisurance sẽ được sử dụng làm ví dụ minh họa cho những phần tiếp theo của luận văn. Trong phần tiếp theo chúng ta sẽ đi sâu vào tìm hiểu các khái niệm chính trong từng tầng của ngôn ngữ Archimate.

* + 1. **Tầng Nghiệp Vụ**
* Meta Model



1. Metamodel của tầng nghiệp vụ

Ở cấp độ trừu tượng hoá các tầng trong ngôn ngữ ArchiMate đều có các khía cạnh thể hiện rất giống nhau, tầng nghiệp vụ cũng được phân biệt ở các khía cạnh sau:

* Khía cạnh cấu trúc:
  + Khía cạnh chủ động
  + Khía cạnh bị động hay khía cạnh dữ liệu
* Khía cạnh ứng xử
* Khía cạnh thông tin
* Những Khái niệm thuộc về cấu trúc

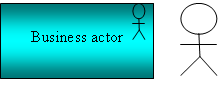
*Khía cạnh chủ động:*

Là những đối tượng thực hiện những ứng xử như *business process*, *business function* trong hệ thống ví dụ như: *business actor*, *business role*, *business collaboration*, *business interface*.

**Business Actor**– Là một thực thể chủ độngthực hiện những ứng xử trong hệ thống

Khía cạnh cấu trúc của tầng nghiệp vụ thường quy đến kết cấu của tổ chức, khái niệm trung tâm của khía cạnh cấu trúc là business actor. Một business actor có thể là một người nào đó ( khách hàng hoặc nhân viên) nhưng cũng có thể là một phòng ban, một nhóm người và tài nguyên của một tổ chức.

Kí hiệu:



Đặc trưng của business actor :

* Một business actor có thể được gán với một hay nhiều business role
* Tên của một business actor nên được đặt là danh từ

**Business Role**– Phát biểu một ứng xử nào đó được thực hiện bởi một business actor đảm nhiệm vai tròđó.

Trong tầng nghiệp vụ, để tăng sự linh động trong mối liên kết giữa những actor và những ứng xử, ArchiMate đưa ra khái niệm trung gian là business role. Nó phù hợp với ý kiến rằng một công việc do một actor thực hiện bên trong một tổ chức luôn được dựa trên một vai trò nào đó do actor đảm nhận. Về mặt này, có ít nhất hai lý do để sử dụng business role, thứ nhất, việc sử dụng một tập những vai trò để mô tả một tổ chức sẽ giúp tăng tính ổn định hơn nhiều so với việc sử dụng những actor đảm nhiệm những vai trò đó. Thứ hai, một vai trò có thể do nhiều actor đảm nhiệm, và ngược lại, một actor có thể đảm nhiệm nhiều vai trò. Business role có thể giúp phân biệt trách nhiệm rõ ràng của từng actor trong tổ chức.

Ký hiệu:



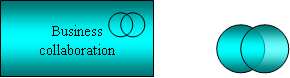
Đặc trưng của business role:

* Business role được gán với một business actor
* Một business role có thể được gán với một hay nhiều business function, business process, business event
* Một business role có thể sử dụng business interface
* Tên của business role nên được đặt là danh từ

**Business Collaboration** – Là một tập những vai trò bên trong một tổ chức, tham gia thực viện một ứng xử cộng tác nào đó.

Những mô tả về kiến trúc tập trung chủ yếu vào kết cấu, điều này có nghĩa là việc thể hiện những mối quan hệ bên trong một tổ chức đóng vai trò rất quan trọng. Để làm sáng tỏ điều này, khái niệm business collaboration được đưa ra. Thông thường, một business process, hay một business function được xác định rõ ràng là một ứng xử bên trong và được gán với một business role, như vậy, trong trường hợp một ứng xử là một tập của nhiều business role thì business collaboration là kết quả của điều này. Một business collaboration là một tập những vai trò,thực hiện một tập những ứng xử bên trong một tổ chức. Không tóm lượt một tập các vai trò của một tổ chức như phòng ban, một business collaboration không có một địa vị chức vụ nào trong tổ chức, nó chỉ nhằm thể hiện một hay một tập những tương tác giữa những vai trò. Nó có thể được xem là một “vai trò ảo”

Ký hiệu:



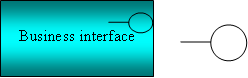
Đặc trưng của business collaboration

* Business collaboration có thể bao gồm một tập những business role
* Business collaboration có thể gán với một hay nhiều business interaction
* Business collaboration có thể sử dụng business interface và application interface
* Business collaboration có thể có nhiều business interface
* Tên của business collaboration nên được khai báo là danh từ

**Business Interface**– Là nơi (vật lý và luận lý) mà những dịch vụ được đưa ra bởi business role có thể cung cấp ra bên ngoài.

Một business interface được giới thiệu để giúp làm sáng tỏ mô hình, nó thể hiện cái kênh hay khu vực mà business service có thể được sử dụng bởi thế giới bên ngoài. Những dịch vụ khác nhau có thể được đưa ra thông qua một số những business interface khác nhau ví dụ như thông qua thư điện tử, điện thoại, hoặc internet.

Ký hiệu:



Đặc trưng của business interface:

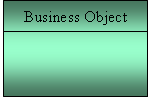
* Business interface có thể được sử dụng bởi business role
* Business interface có thể được gán với một hay nhiều business service, nghĩa là những dịch dụ này có thể được đưa ra bên ngoài thông qua interface
* Tên của business interface nên được đặt là danh từ

*Khía cạnh passive:*

Là những đối tượng được vận dụng bởi những ứng xử như business process, business function, business service…Nó miêu tả những khái niệm và những thông tin quan trọng về nghiệp vụ

**Business Object**– Là những thực thể bị động được vận dụng bởi những ứng xử như là business function, business process, nó là một đơn vị thông tin phù hợp về khía cạnh nghiệp vụ.

Ký hiệu



Đặc trưng về business object:

* Business object có thể được truy xuất bởi business function, business process, business event, business service,
* Một business object có thể hiện thực, kế thừa và mở rộng, kết hợp với những business object khác
* Một business object có thể được hiện thực bởi presentation hay data object
* Tên của business object nên được thể hiện là danh từ
* Những khái niệm thuộc về ứng xử

*Ứng xử bên ngoài*

**Business service**– là mảng chức năng cung cấp cho môi trường, nó độc lập với cái cách mà các những chức năng này được hiện thực bên trong.

Business service được sử dụng để đưa những chức năng thuộc về nghiệp vụ ra thế giới bên ngoài. Cần phân biệt giữa những business service hướng bên ngoài (hướng tới cung cấp dịch vụ cho khách hàng) và business service hướng bên trong (hướng cung cấp những chức năng cho business process hoặc business function bên trong một tổ chức).

Ký hiệu:



Những đặc trưng về business service

* Business service đưa chức năng của những business role và business collaboration cho môi trường
* Những chức năng này được truy xuất thông qua một hoặc nhiều business interface.
* Một business service được hiện thực bởi một hay nhiều business process, business function, business interaction.
* Business service có thể truy xuất business object
* Một business service có thể bao gồm nhiều business service khác và có thể sử dụng những business service khác.
* Business service có thể kết hợp với một value
* Business service có thể được sử dụng bởi business function, business process, hoặc business interaction
* Tên của business service có thể được thể hiện bằng động từ thêm “-ing” hay thêm cụm từ “service”.

*Ứng xử bên trong*

Bên trong tổ chức, dịch vụ của tổ chức được hiện thực bởi những ứng xử nghiệp vụ, chúng ta có một số khái niệm: business process, business function, business activity, business interaction.

Business activity, business process, business function là đơn vị ứng xử bên trong được thực hiện bởi một hay nhiều business role bên trong một tổ chức.

**Business Activity -** Là cấp độ nhỏ nhất của ứng xử thuộc về nghiệp vụ

Trong một số ngôn ngữ khác Business activity còn được gọi là nhiệm vụ, một business activity không thể bị phân nhỏ hơn

Ký hiệu



**Business process**–Là một đơn vị ứng xử bên trong hoặc một tập những ứng xử bên trong có mối quan hệ “nguyên nhân hệ quả” với nhau để đưa ra một tập những thành phẩm và dịch vụ

Business process miêu tả luồng công việc hay luồng giá trị,có thể bao gồm những business process hay những business function nhỏ hơn. Nó có điểm bắt đầu và kết quả hay cách thức đạt được kết quả. Đôi khi, nó được mô tả là “từ khách hàng đến khách hàng” và những khách hàng này lànhững khách hàng thuộc bên trong, hay nói cách khác những khách hàng này là những quy trình nhỏ bên trong một tổ chức. Giữa business process và business function có mối quan hệ ngầm “nhiều-nhiều”, ví dụ process miêu tả một luồng của những activity, function lại nhóm những activity lại theo một số tiêu chí như tài nguyên, kiến thức, kỹ năng …

Ký hiệu



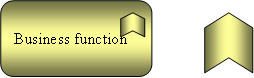
Những đặc trưng về Business Process:

* Một business process có thể kích hoạt hoặc bị kích hoạt bởi những đối tượng thuộc khía cạnh ứng xử thuộc về nghiệp vụ như: business event, function, interaction hay business process khác
* Một business process có thể truy xuất business object
* Business process có thể hiện thực một hay nhiều business service, và nó có thể sử dụng những business service khác cũng như là application service
* Tên của business process nên được đặt là một động từ ở thì hiện tại.

**Business Function**– Là đơn vị ứng xử bên trong, nó nhóm những ứng xử liên quan với nhau, ví dụ về yêu cầu kỹ năng, kiến thức, tài nguyên…và được thực hiện bởi một vai trò bên trong tổ chức.

Business function quy về chức năng có thể hữu ích cho business process, nó nhóm những ứng xử dựa trên các tiêu chí như : năng lực, tài nguyên, kỹ năng.Trong khi business process của tổ chức được định nghĩa dựa trên giá thành phẩm và dịch vụ mà tổ chức đó quan tâm thì business function được dựa trên mối quan hệ tài nguyên đến nhiệm vụ, và ứng dụng hỗ trợ.

Ký hiệu



Đặc trưng về business function:

Một business function có thể kích hoạt hoặc bị kích hoạt bởi những đối tượng thuộc khía cạnh business behavior như: business event, process, interaction hay business function khác

* Một business function có thể truy xuất business object
* Business process có thể hiện thực một hay nhiều business service, và nó có thể sử dụng những business service khác cũng như là application service
* Businsess function có thể được gán với role hay application component.
* Tên của business function nên được đặt là một động từ ở thì hiện tại

**Business interaction**– Được định nghĩa là một đơn vị ứng xử biểu diễn sự kết hợp của hai hay nhiều business role

Cũng giống như business function hay business process, business interaction cũng là một đơn vị ứng xử, nhưng nó được thực hiện bởi sự kết hợp của hai hay nhiều vai trò trong một tổ chức, tương tự như business function hay business process, business interaction cung cấp chức kết quả ra môi trường thông qua một business service.

Kí hiệu minh hoạ:



Những đặc trưng của business interaction

* Một business interaction có thể kích hoạt hay được kích hoạt bởi các ứng xử khác(business event, business function, business process…)
* Business interaction có thể truy xuất business object
* Business interaction có thể hiện thực một hay nhiều business service
* Business interaction có thể sử dụng business service hay application service khác
* Business interaction được gán với business collaboration hay application collaboration
* Tên của một Business interaction nên được khai báo là động từ thì hiện tại

**Business Event**–Là một điều gì đó diễn ra ở bên ngoài và tác động lên business process, business function, business interaction.

Business event biểu diễn một điều gì đó diễn ra(bên ngoài) có thể tác động lên business fucntion, business process, business interaction. Một business event thường được dùng để mô hình điều gì gây nên business function, business process, ngoài ra còn có một số kiểu khác của business event có thể được dùng như: sự kiện xảy ra dẫn đến ngắtmột business process.

Ký hiệu



Đặc trưng của business event:

* Một business event có thể gây ra hoặc được gây ra bởi business process, business function, business interaction
* Business event có thể truy xuất business object
* Business event có thể bao gồm những business event khác
* Tên của business event nên được đặt là động từ trong thì hoàn thành
* Những khái niệm thuộc về thông tin

Trái ngược với khía cạnh cấu trúc và khía cạnh ứng xử - khía cạnh liên quan chủ yếu đến góc nhìn chức năngtrong enterprise. Khía cạnh thông tin tập trung vào góc nhìn mà chúng ta gọi là “góc nhìn có chủ đích”. Chúng cung cấp cái cách để liên kết giữa khía cạnh chức năng của tổ chức với mục đích kinh tế. Nó đưa ra những thành phẩm mà tổ chức đem đến cho khách hàng.

**Presentation** – Là dạng thông tin có thể đọc được, được mang bởi business object như là một tài liệu.

Presentation, ví dụ là những thông điệp hay tài liệu, là đối tượng mang những thông tin có thể đọc được, những thông tin này có mối quan hệ chặt chẽ với business object. Presentation có thể được phân loại theo một số cách, ví dụ, theo môi trường truyền thông (giấy,điện,âm thanh…) hay theo định dạng(HTML, Text, ASCII…)

Ký hiệu:



Đặc trưng về Presentation:

* Presentation có thể hiện thực một business object
* Presentation có thể kết hợp với meanning
* Một presentation luôn đi kèm với duy nhất một business object, một business object có thể có nhiều representation
* Tên của Presentation nên được thể hiện là danh từ

**Meanning** – Là sự đóng góp(của presentation) của business object vào kiến thức hay ý kiến chuyên môn của một vài actor trong một hoàn cảnh đặc biệt nào đó

Meanning là một sự đóng góp về kiến thức hay kiến thức mở rộng trong một ngữ cảnh đặc biệt mà presentation của một data object dành cho một actor nào đó. Hay nói cách khác, meanning là đối tượng mang giá trị thông tin của một business object đế cho một user trong một hoàn cảnh cụ thể. Nó đưa ra một sự miêu tả thông tin chính xác, chắc chắn đến một người dùng hay một loại người dùng cụ thể nào đó.

Ký hiệu



Đặc trưng của meanning:

* Một meanning có thể kết hợp với một presentation
* Tên của meanning nên được thể hiện là danh từ hoặc cụm danh từ

**Value**− Là những thứ giúp những nhóm người liên quan đánh giá đúng một sản phẩm hay một dịch vụ.

Khái niệm Value của một thành phẩm hay một dịch vụ sẽ giúp những người liên quan (nhà cung cấp,khách hàng) đánh giá đúng giá trị của chúng mang lại, khái niệm này có thể giúp chúng được đánh giá bởi những người cung cấp, nhưng thường là hỗ trợ việc đánh giá cho những người yêu cầu những dịch vụ này ví dụ như khách hàng. Value có thể đi theo hai cách: nó có thể là những gì mà nhà cung cấp đạt được khi “bán” hay nói cách khác là đưa ra những thành phẩm hay những dịch vụ này. Hay nó là những gì mà những khách hàng đạt được khi nhận hay nói cách khác là “mua” những thành phẩm hay dịch vụ đó.

Ký hiệu:



Đặc trưng của value:

* Value có thể kết hợp với những dịch vụ, những sản phẩm (nó có thể là một phần của sản phẩm), những business role hoặc actor sử dụng chúng.
* Value có thể chứa những sub-value
* Mặc dù tên value có thể được thể hiện bằng nhiều cách, nhưng tên của value thuộc về chức năng của một dịch vụ thì nên được đặt như là một hành động, một kết quả đạt được

**Product**– Được định nghĩa là một tập hợp của những dịch vụ đi cùng với bản quy định về những đặc trưng, quyền lợi, và yêu cầu sử dụng những dịch vụ này

Một product được xem như là một tập của những service, cùng với những quy tắt để sử dụng chúng. Những service này thường là những service thuộc về tổ chức, những application service cũng có thể là một phần của product. “Mua” một product sẽ cho khách hàng quyền để sử dụng những service kết hợp với nó.

Ký hiệu:



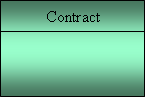
Đặc trưngcủa Product:

* Một product có thể chứa những business service, application service và một contract.
* Một product có thể kết hợp với một value
* Tên của product có thể là tên thường được sử dụng trong quá trình truyền thông với khách hàng, hay thường được khai báo là một danh từ tổng quát

**Contract -** Là những quy tắc mang tính hình thức hoặc không chính thức của một thoả thuận quy định về quyền và nghĩa vụ kết hợp với một product

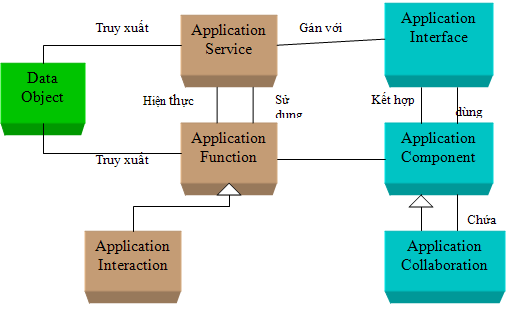
Khái niệm contract thường được sử dụng để mô hình một “bản hợp đồng” mang ý nghĩa pháp lý, không những thế, nó còn thể hiện những sự đồng thuận chưa chính thức về một product. Nó có thể bao gồm, hoặc có thể là một Service Level AgreeMent(SLA), nó miêu tả những thoả thuận về chức năng và chất lượng về những dịch vụ của product.

Ký hiệu:



Đặc trưng của Contract:

* Một contract là sự kế thừa và mở rộng của business object, những mối quan hệ áp dụng business object cũng có thể áp dụng cho contract.
* Một contract có thể chứa những contract khác
* Một product có thể chứa một contract
* Tên của contract nên được đặt là danh từ
  + 1. **Tầng Ứng Dụng**
* Meta Model



1. Meta model của tầng ứng dụng

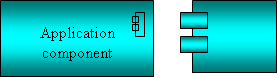
Ở cấp độ trừu tượng hoácác tầng trong ngôn ngữ ArchiMate đều có chung các khía cạnh thể hiện:

* Khía cạnh cấu trúc:
  + Khía cạnh chủ động
  + Khía cạnh bị động hay khía cạnh dữ liệu
* Khía cạnh ứng xử
* Những khái niệm thuộc về cấu trúc

**Application Component -** Một application component được định nghĩa như là một thành phần có thể triển khai độc lập, và có thể thay thế được của hệ thống. Nó tóm lược nội dung bên trong nó, và cung cấp chức năng ra bên ngoài thông qua một tập giao diện.

Khái niệm chính quan trọng nhất trong khía cạnh cấu trúc là application component. Khái niệm được sử dụng để mô hình và kiến trúc mọi thực thể thuộc về khía cạnh cấu trúc trong tầng ứng dụng: những thực thể này không chỉ là thành phần của phần mềm,một phần của ứng dụng mà nó còn có thể là một phần mềm ứng dụng hoàn chỉnh, hay một ứng dụng con, một hệ thống thông tin.

Kí hiệu:



Những đặc trưng củaapplication component:

* Một applictioncomponent có một hay nhiều application interface
* Một data object được tạo, truy xuất, thay đổi nội dung hay bị huỷ bởi một appliction component hay application function.
* Application component có thể là một phần của appliction collaboration
* Application component có thể bao gồm nhiều application component khác
* Một appliction component có thể được gán với một hay nhiều application function, business process, business function
* Một luồng dữ liệu có thể tồn tại giữa các application component
* Tên của một application component nên được khai báo là một danh từ.

**Application Collaboration -**  Được định nghĩa như là một dạng hình thể tạm thời của hai hay nhiều application component cùng tham gia vào một hay nhiều quá trình tương tác ứng dụng.

Cũng như tầng nghiệp vụ, trong kiến trúc ứng dụng, mối quan hệ giữa các application component là thành phần cơ bản nhất, vì thế ArchiMate giới thiệu một khái niệm là application collaboration, nó được xem như là một tập các application component tham gia vào một tương tác, khái niệm này cũng tương tự như collaboration được định nghĩa trong UML 2.0

Kí hiệu:



Những đặc trưng về application collaboration:

* Một application collaboration chứa hai hay nhiều application component.
* Một application collaboration có thể được gán với một hay nhiều application interaction
* Một application collaboration có thể sử dụng application interface và ngược lại.
* Tên của application collaboration nên được khai báo là danh từ

**Application Interface -** định nghĩa một tập các chức năng và sự kiện được cung cấp bởi application component hay những chức năng và sự kiện được yêu cầu bởi môi trường.

Application interface được xem như là một kênh mà thông qua nó, những dịch vụ của một application component có thể được truy xuất. Ở ý nghĩa rộng hơn, một application interface định nghĩa một đặc trưng cơ bản thuộc về ứng xử: như một tập những chức năng hoặc sự kiện được cung cấp bởi appliction component hay được yêu cầu từ bên ngoài, vì vậy, nó được sử dụng để miêu tả những thành phần mang tính chức năng của ứng dụng. Có sự phân biệt giữa *interface cung cấp* và *interface yêu cầu*. Application interface cũng có thể được sử dụng để mô hình giao diện “ứng dụng đến ứng dụng” (đưa ra những application service hướng bên trong) và “ứng dụng đến nghiệp vụ” (đưa ra những application service hướng ra bên ngoài)

Ký hiệu:



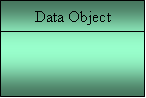
Những đặc trưng của application interface:

* Application interface có thể được gán với nhiều application service hay business service, điều này có nghĩa là application interface cung cấp những dịch vụ này ra môi trường
* Tên của application interface nên được khai báo là danh từ

**Data Object -** Được định nghĩa là đối tượng chứa thông tin rõ ràng, mạch lạc, và phù hợp cho quá trình xử lý.

Trong tầng ứng dụng, chúng ta cũng phân biệt khía cạnh bị động của một thành phần được gọi là data object.

Ký hiệu:



Đặc trưng của Data Object:

* Data object có thể được tạo,xoá,đọc nội dung bởi application component, application service hay application function
* Một artifact hiện thực một data object
* Một data object có thể bao gồm nhiều data object khác
* Một data object có thể được kế thừa và mở rộng
* Một data object nên được khai báo là một danh từ
* Những khái niệm thuộc về ứng xử

**Application Service -** Được định nghĩa là một đơn vị chức năng có thể nhìn thấy được từ bên ngoài, nó được cung cấp bởi một hay nhiều thành phần ứng dụng, được đưa ra thông qua những giao diện được định nghĩa rõ ràng, và có ý nghĩa với môi trường.

Khái niệm application service cung cấp cách mô tả rõ ràng những chức năng mà thành phần ứng dụng chia sẽ với nhau và cung cấp ra môi trường bên ngoài. Internal service được đưa ra thông qua giao diện “*ứng dụng đến ứng dung”*. External service có thể được đưa ra thông qua giao diện *“ứng dụng đến nghiệp vụ”*. Application service đưa application function ra môi trường.

Ký hiệu:



Những đặc trưng của application service:

* Một application service có được hiện thực bởi một application component, hay một application function
* Một application service có thể truy xuất một hay nhiều data object
* Một application service có thể được sử dụng bởi application component
* Một application service có thể chứa nhiều application service của các application component khác, và có thể sử dụng application service khác.
* Một application service có thể được truy sử dụng bởi business process, business function, business interaction
* Tên của application service có thể được khai báo bằng cách thêm “-ing”(nếu đặt theo Tiếng Anh) vào cuối hoặc thêm từ “dịch vụ”.

**Application Function -** Là ứng xử bên trong của một application component, nó hiện thực một hay nhiều application service

Application function là một đơn vị ứng xử bên trong của một application component, application function được đòi hỏi hiện thực một hay nhiều application service. Và lưu ý, trong mô tả kiến trúc không nên thể hiện chi tiết quá nhiều ứng xử bên trong của application component, vì có thể sẽ đụng độ với quá trình thiết kế chi tiết.

Ký hiệu:



Những đặc trưng về application function:

* Một application function hiện thực một hay nhiều application service.
* Một application function có thể truy xuất data object
* Một application function có thể sử dụng service của các application function khác, cũng như là của tầng kỹ thuật
* Tên của application function nên được khai báo là động từ (thêm “-ing” ở cuối nếu đặt theo Tiếng Anh).

**Application Interaction-** Là ứng xử sinh ra bởi sự kết hợp của hai hay nhiều application component.

Những ứng xử tương tác của những application component được gọi là application interaction

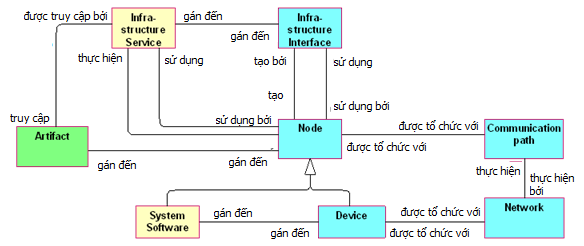
Ký hiệu:



Tổng quát Application interaction:

* Application interaction có thể được gán với một application collaboration
* Application interaction có thể hiện thực một application service
* Application interaction có thể sử dụng những dịch vụcủa những application component và infrastructure service.
* Application interaction có thể truy xuất data object.
* Application interaction nên được khai báo là động từ
  + 1. **Tầng Kỹ thuật**
* Các thành phần trong tầng kỹ thuật

Hình bên dưới cho ta một cái nhìn khái quát các thành phần trong tầng kỹ thuật và mối quan hệ giữa các thành phần với nhau.

****

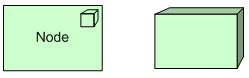
1. Metamodel Tầng kỹ thuật

* Khái niệm thuộc về cấu trúc

Cấu trúc của tầng kỹ thuật được mô hình sử dụng những khái niệm node, device, và infrastructure interface.

N**ode** - là một thể hiện của thực thể có cấu trúc trong tầng kỹ thuật. Ví dụ như “application server”,“database server”, “UNIX server”, “firewall”, “mail server” and “web server”. Sử dụng node, chúng ta có thể miêu tả cả tài nguyên máy tính vật lý (máy tính), cũng như phần mềm (hệ điều hành, phần mềm ứng dụng) chức năng.

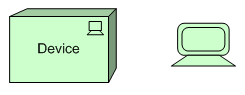
Ký hiệu:



**Device**- là một sự kế thừa và mở rộng của node, nó là một tài nguyên máy tính vật lý, ví dụ như mainframe, PCs, router, và “load banlancers”.Một device là một thiết bị vật lý mà dựa trên nó những artifact được triển khai để thực thi.

Nói chung, một node bao gồm trong số các nguồn tài nguyên vật lý (device) và system software

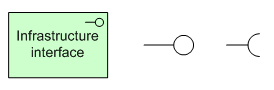
Ký hiệu:



**Infrastructureinterface** (còn được gọi là technology-interface) - cho thấy cách node cung cấp chức năng ra với môi trường. Một infrastructure interface được xem như là một *điểm truy xuất*, nơi mà những dịch vụ của một node có thể được truy xuất bởi những node khác haynhững application component và application colloboration.

Bạn có thể mô tả các thỏa thuận và điều kiện,cả hai bên phải tuân theo để sử dụng các chức năng của nhau.

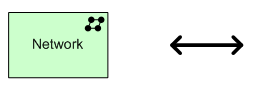
Ký hiệu:



**Network**– Được định nghĩa như một môi trường truyền thông vật lý giữa hai hay nhiều device.

Một networklà một phương tiện truyền thông vật lý hay luận lý giữa hai hoặc nhiều node hoặc các thiết bị, và nhận ra các đường dẫn truyền thông. Ví dụ: mạng không dây, Ethernet, Surfnet, mạng nội bộ, internet. Network thường là một đường liên kết giữa hai devide. Network có những thuộc tính như băng thông, độ trễ. Network thể hiện sự hiện thực vật lý của communication path giữa những node.

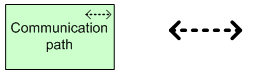
Ký hiệu:



**Communication path** - là liên kết giữa hai hay nhiều node, thông qua communication pathnhững node có thể trao đổi thông tin với nhau.

Communication path được sử dụng để mô hình mối quan hệ truyền thông giữa những node. Nó được hiện thực bởi một hay nhiều network.

Ký hiệu:



* Khái niệm thuộc về ứng xử

**System software** –thể hiệnmột môi trường phần mềm cho các thành phần và các đối tượng có thể được triển khaidưới hình thức những artifact.

Trái ngược với một device, system software là một sự trừu tượng của tài nguyên vật lý. Ví dụ như hệ điều hành, hệ thống database, CORBA, J2EE và phần mềm cho ứng dụng server.

Ký hiệu



I**nfrastructure service** - được định nghĩa là một đơn vị chức năng có thể được nhìn thấy được từ bên ngoài, có ý nghĩa đối với môi trường, nó được cung cấp bởi một hoặc nhiều node và đưa ra thông qua những infrastrure interface được định nghĩa rõ ràng.

Ký hiệu



* Khái niệm thuộc về thông tin

**Artifact** - là 1 khái niệm cấu trúc bị động, nó mô hình thông tin hoặc dữ liệu, tương tự như data object và business object. Ngôn ngữ ArchiMate định nghĩa artifact như là một mảnh thông tin vật lý được sử dụng (hoặc được cung cấp) bởi một quá trình phát triển phần mềm hoặc bởi sự triển khai và hoạt động của hệ thống.

Trong thực tế, một artifact là một thể hiện cụ thể trong thế giới vật lý.Artifact thường được dùng để mô hình những sản phẩm (phần mềm) ví dụ như là những bảng cơ sở dữ liệu, source files, scripts, executables, tài liệu.

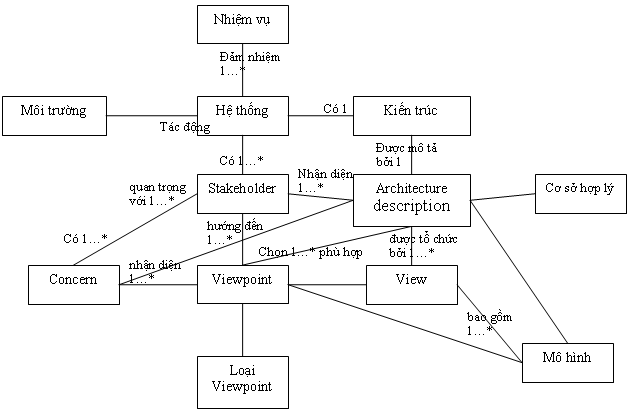
Ký hiệu



* + 1. **Viewpoint**
* Viewpoint là gì-Tại sao lại cần sử dụng Viewpoint

Việc xậy dựng, duy trì và phát triển một kiến trúc enterprise là một nhiệm vụ rất khó khăn và phức tạp, bởi vì nó liên quan đến nhiều người khác nhau với những kiến thức và kinh nghiệm khác nhau trong việc sử dụng những chú thích cũng như là ký hiệu. Cũng chính điều này dẫn đến những trở ngại trong việc truyền thông giữa những stakeholder, khó khăn trong việc thống nhất cũng như là nắm rõ được architecture description. Điều này dẫn đến cần đưa ra một giải pháp để những stakeholder và những kiến trúc sư có thể truyền thông với nhau hiệu quả bằng cách đưa ra được những view khác nhau về architecture description mà họ có thể hiểu được. Vì vậy, để hỗ trợ giải quyết vấn đề phức tạp trên, viewpoint được đưa ra như là một giải pháp thiết thực.

Viewpoint trừu tượng hoá một tập những mô hình để thể hiện một kiến trúc enterprise, mỗi một tập mô hình trên đều hướng về một loại stakeholder với những mối quan tâm cụ thể. Viewpoint có thể được sử dụng để mô tả những khía cạnh độc lập, cũng như là mô tả mối quan hệ giữa hai hay nhiều khía cạnh với nhau.



1. Mô hình khái niệm về architecture description (IEEE Computer Society, 2000)

Theo mô hình khái niệm trên cho thấy, view và viewpoint đóng vai trò trung tâm như là một cách thức chuẩn để mô tả kiến trúc(architecture). Ở những mục tiếp theo, sẽ cung cấp cho chúng ta những đặc điểm chi tiết cũng như cách tạo, các sử dụng, các chọn lựa viewpoint cho hợp lý. Bên cạnh đó chúng ta cũng sẽ được giới thiệu những framework phổ biến về viewpoint.

* Khái niệm Viewpoint

**Nguồn gốc hình thành và phát triển của Viewpoint**

Khái niệm Viewpoint không phải là một khái niệm mới mẻ, từ giữa những năm 1980, Multiview(Wood-Harper et al. 1985) đã được giới thiệu, Multiview định ra năm viewpoint cho quá trình phát triển hệ thống thông tin:

* Human Activity System
* Information Modelling
* Socio-Technical System
* Human-Computer Interface
* The Technical System

Song song với sự phát triển của Multiview, CRIS Task Group of IFIP Working Group cũng phát triển những khái niệm tương tự, những view dành cho stakeholder cũng được thống nhất thông qua những “sự thể hiện” phù hợp.

Khái niệm Viewpoint không chỉ giới hạn chỉ đối với cộng đồng hệ thống thông tinmà nó còn được giới thiệu bởi cộng đồng phát triển phần mềm. Năm 1990, những nhà nghiên cứu phần mềm đã đưa ra những giải pháp cho vấn để “the multiple perspectives problem”(Finkelstein etal. 1992, Kotonya and Sommerville 1992, Nuseibeh 1994, Reeves et al. 1995), họ đã tập trung vào nghiên cứu giải quyết vấn đề làm thế nào để tổ chức và hướng dẫn phát triển phần mềm với nhiều actor khác nhau về kiến thức, kinh nghiệm, và chiến lược phát triển. Kết quả là một framework tổng quát đã được ra đời và phát triển(Finkelstein et al. 1992, Kotonya and Sommerville 1992, Nuseibeh 1994), trong framework này viewpoint kết hợp với những khái niệm như *actor, role,* hay *agent*, và sáng kiến được đưa ra cho rằng *actor* nắm giữ những góc nhìn và những view. Những sáng kiến ban đầu của “xây dựng phần mềm hướng viewpoint” này đã được đưa vào chuẩn IEEE-1471 dành cho architecture description(IEEE Computer Society, 2000). Và định nghĩa của chúng ta về viewpoint dựa trên chuẩn này.

*Định nghĩa:****Viewpoint là sự quy định rõ ràng của những thoả thuận về việc xây dựng và sử dụng những view.***

Trong ngữ cảnh kiến trúc, viewpoint cung cấp cách thức để tập trung vào một khía cạnh cụ thể của một architecture description. Những khía cạnh này được xác định bởi những mối quan tâm của stakeholder với tất cả những người tham gia vào việc truyền thông. Viewpoint được thiết kế cho mục đích phục vụ như là một phương tiện trong quá trình truyền thông của một cuộc hội thảovề một khía cạnh cụ thể nào đó của một kiến trúc. Điều gì được và không được thể hiện trong một view đều phụ thuộc vào phạm vi của viewpoint và dựa trên mối quan tâm của những stakeholder.

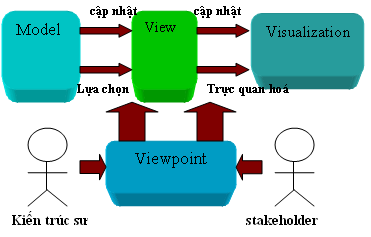
* Model, View, Visualization

Để có thể thiết kế, chọn lựa, và sử dụng những viewpoint một cách đúng đắng, trước hết chúng ta cần phải nắm giữ được những khái niệm và những nguyên tắt quan trọng liên quan đến model, view, visualization.

Một trong những nguyên tắc quan trong trong hướng tiếp cận của chúng ta cần tách biệt giữa *nội dung*và *sựthể hiện*hay *sự trực quan* của một view. Điều này giúp chúng ta có thể sử dụng nhiều kỹ thuật trực quan khác nhau cho cùng một mô hình

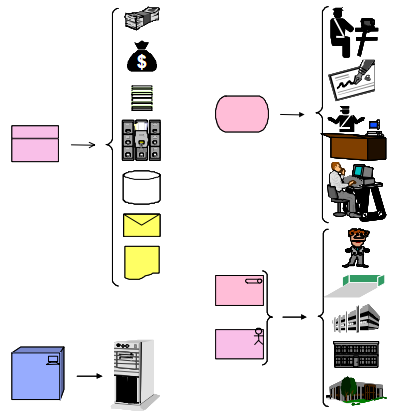
* *Nội dung:* Thường được gọi là view, view là sự lựa chọn bắt nguồn từ mô hình hình học của một kiến trúc, và được thể hiện ở những dạng tương tự như những khái niệm được dùng để mô hình.
* *Sự thể hiện:* Thường được gọi là visualization, nó có thể có nhiều định dạng, từ những lược đồ chuẩn cho đến những ‘bảng’, những ‘tranh vẽ’, hoặc một ‘đoạn phim’,chỉ cốt để cung cấp một cái nhìn trực quan, dễ hiểu về một mô hình nào đó của kiến trúc cho những stakeholder.

Việc thay đổi một chức năng nào đó ở góc độ trực quan cũng đều có thể dẫn đến việc cập nhật sự thay đổi của view và của mô hình bên dưới:



1. Mối quan hệ giữa Model-view-visualization

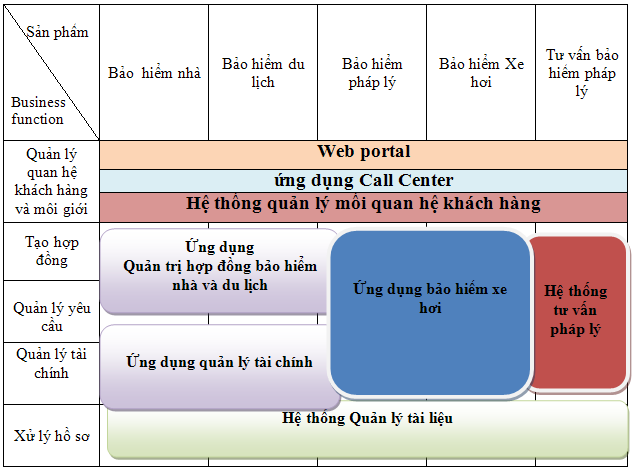
Để minh hoạ cho sự khác biệt giữa view và visualization chúng ta có thể xem hình minh hoạ bên dưới:



1. Minh hoạ của visualization từ những “archimate shape”

Với hình minh hoạ trên, ta thấy được một số những nguyên tắt *sự thể hiện* có thể được áp dụng vào “trực quan hoá mô hình”. Thấy được rằng những nguyên tắt này nhấn mạnh điều quan trọng trong *sự trực quan hoá* một view nào đó là có thể tìm được những ký hiệu tượng trưng sinh động, phù hợp và trực quan có thể thay thế những ký hiệu ArchiMate.

Để minh hoạ rõ ràng hơn về sự khác biệt giữa một model, một view,và visualization của nó chúng ta đưa ra ví dụ về *landscape map viewpoint*. Landscape map, là một kỹ thuật để trực quan hoá kiến trúc enterprise. Chúng thể hiện những phần tử ở dạng ‘map’ 2D dễ hiểu,view thể hiện bằng Landscape map sẽ cung cấp cho những người quản lý, những stakeholder không có kiến thức nhiều về kỹ thuật một cái nhìn tổng quát ở mức high-level của một kiến trúc, mà không buộc họ phải hiểu biết về những kỹ thuật chuyên môn về bản vẽ của một kiến trúc.



1. Lanscape map minh hoạ cho hệ thống bảo hiểm ArchiSurance

Hình trên là một ví dụ rõ ràng về landscape map, ta thấy rằng đối với một hệ thống với rất nhiều quy trình, đưa ra những thành phẩm và những dịch vụđa dạng có thể dẫn đến một bản vẽ với nhiều quá nhiều chi tiết được thể hiện trong một hình. Điều này dẫn đến một sự phức tạp và rối rắm đối với người xem. Vì thế, landscape map viewpoint chính là sự hỗ trợ đắc lực để giải quyết vấn đề trên. Hình trên là một bức tranh sinh động thể hiện những chức năngmà hệ thống bảo hiểm ArchiSurance cung cấp bằng việc đưa ra những sản phẩm được hiện thực bởi những application component. Rõ ràng, landscape map là một thể hiện sinh động, rõ ràng về một hệ thống thông tin bằng những ký hiệu tượng trưng trực quan được chuyển đổi từ những “ArchiMate shape” mà không thực sự chứa một ký hiệu ArchiMate nào.

Trong ngữ cảnh kiến trúc một hệ thống, sự khác biệt cũng như mối quan hệ giữa model, view và visualization đóng vai trò rất quan trọng trong việc truyền thông giữa những những stakeholder và những kiến trúc sư. Vì vậy việc nắm bắt những khái niệm này là một đòi hỏi cấp thiết.

* Xây dựng, lựa chọnvàsử dụng Viewpoint

Việc tạo, chọn lựa và sử dụng viewpoint là nội dung chính yếu và quan trọng khi nói về viewpoint. Như ta biết, một trong những khó khăn của kiến trúc sư là việc xác định được đâu là những viewpoint cần được sử dụng. Việc xác định này khá phức tạp, bởi nó liên quan đến nhiều stakeholder với nhiều lĩnh vực và nhiều mối quan tâm khác nhau. Thực tế, đã có những framework ra đời để hỗ trợ cho kiến trúc sư giải quyết những vấn đề này như Zachman Framework, Kruchten’s 4+1 (Kruchten 1995), RM-ODP (ITU 1996), và TOGAF (The Open Group 2002). Bên cạnh đó, nhiều tổ chức lớn cũng đã thiết kế cho họ những framework về viewpoint để mô tả kiến trúc của họ. Những framework này chúng ta sẽ được giới thiệu ở những phần tiếp theo. Song, việc sử dụng những framework sẵn có mặc dù sẽ mang đến cho chúng ta những sự dễ dàng nhất định, nhưng “chọn lựa nhẹ nhàng” này sẽ không mang đến cho chúng ta cách nhìn vào bên trong vấn đề, không thấy được lý do và động lực tại sao phải chọn những viewpoint. Vì vậy, trong phần này chúng ta sẽ được đi vào tìm hiểu các loại viewpoint để rút ra được kết luận đúng đắn về việc tạo, chọn lựa và sử dụng những viewpoint nào là hợp lý trong từng ngữ cảnh cụ thể.

* Phân loại Viewpoint

Như đã được đề cập trước, trong ngữ cảnh kiến trúc hệ thống, một kiến trúc sư luôn phải đối mặt với nhiều stakeholder và những mối quan tâm khác nhau của họ, để giúp kiến trúc sư chọn được những viewpoint hợp lý và đúng đắn, người ta đưa ra một framework để định nghĩa và phân loại viewpoint và view, framework này dựa trên hai yếu tố chính đó là *mục đích*và *nội dung*. *Mục đích*là điều mà những viewpoint này phải phục vụ, và *nội dụng*là những gì mà chúng phải thể hiện ra. Dưới đây là ba kiểu kiến trúc được hỗ trợ định nghĩa theo yếu tố*mục đích*:

* *Thiết kế*: Thiết kế những viewpoint hỗ trợ kiến trúc sư và nhà thiết kế trong quá trình thiết kế từ phát thảo cho đến thiết kế chi tiết.
* *Ra quyết định*: Là những view hỗ trợ những nhà quản lýra quyết định bằng cách cung cấp sự thấu hiểu sâu sắc về kiến trúc mối quan hệ giữa những khía cạnh trong hệ thống, thường thông qua phép chiếu và phép giao của mô hình bên dưới.
* *Thông tin*: Những viewpoint này được sử dụng để thông tin với các stakeholder để đạt được sự thấu hiểu, sự đồng tình và thuyết phục đối phương.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Stakeholder đặc trưng** | **Mục đích** | **Ví dụ** |
| ***Thiết kế*** | Kiến trúc sư, kỹ sư phát triển phần mềm, người thiết kế quy trình kinh doanh | Hướng dẫn, hỗ trợ thiết kế, ra quyết định thiết kế | Lược đồ BPMN, UML, flowchart, lược đồ ER |
| ***Ra quyết định*** | Nhà quản lý, CEO, CIO | Hỗ trợ ra quyết định quan trọng | Landscape map, list, report |
| ***Thông tin*** | Khách hàng, nhân viên, và những người khác | Giả thích, thuyết phục đối phương | Phim, ảnh, biểu đồ sinh động |

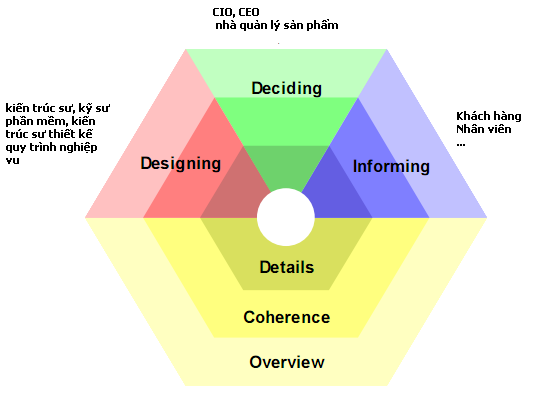
Mục đích của sự phân loại theo *mục đích* này giúp cho những kiến trúc sư tìm thấy được những viewpoint phù hợp với mục đích mà họ cần phục vụ trong một hoàn cảnh nào đó. Điều này có nghĩa là framework này không phải cung cấp một sự phân loại “cứng nhắc” tức không phải mọi viewpoint đều phải được chia làm một trong ba loại, và những loại này không phải đều chỉ được dùng trong một hoàn cảnh cố định nêu trên mà không thể áp dụng trong trường hợp khác. ví dụ viewpoint hỗ trợ ra quyết định không chỉ dành cho những nhà quản lý mà đôi khi nó còn được dùng để thông tin một cách có hiệu quả với các stakeholder khác.

Và đối với *nội dung,* chúng ta định nghĩa theo những cấp độ trừu tượng sau:

* *Chi tiết*: Những view của cấp độ chi tiết thường là bản vẽ chi tiết về một tầng, một khía cạnh nào đó của kiến trúc, những stakeholder ở đây thường là những kỹ sư phần mềm, những người trực tiếp tham gia vào thiết kế và hiện thực những thành phần của phần mềm, hay có thể là người quản lý quy trình, là người chịu trách nhiệm về tính hiệu quả của sự thực thi quy trình.
* *Gắn kết*: Những view ở cấp độ này có thể thể hiện ở nhiều tầng hay nhiều khía cạnh, sự mở rộng ở nhiều tầng, nhiều góc nhìn khác nhau này giúp những stakeholder có thể tập trung vào những mối quan hệ trong kiến trúc. Những stakeholder ở đây thường là nhà quản lý chức năng, những người chịu trách nhiệm về các dịch vụ IT hay quy trình kinh doanh.
* *Khái quát*: Tại cấp độ này, những view thường được thể hiện ở nhiều khía cạnh và nhiều tầng. Những view ở cấp độ khái quát này này thường hướng đến enterprise architect và những người ra quyết định như CEO hay CIO.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Stakeholder đặc trưng** | **Mục đích** | **Ví dụ** |
| **Chi tiết** | Kỹ sư phần mềm, người quản lý quy trình kinh doanh | Thiết kế, quản lý | Lược đồ thể hiện quy trình, lược đồ lớp UML |
| **Gắn kết** | Nhà quản lý chức năng hệ thống | Phân tích sự phụ thuộc, đánh giá tác động của sự thay đổi | Những view thể hiện mối quan hệ như “use, realize, assign” |
| **Khái quát** | Enterprise architect, CEO, CIO | Quản lý sự thay đổi | Landscape map |

Hình bên dưới sẽ miêu tả trực quan về*mục đích*và cấp độ trừu tượng hoá(*nội dung)* của viewpoint:



1. Phân loại viewpoint theo *mục đích và nội dung (*ArchiMate D3.4.1a v2.6)

* Sử dụng Viewpoint

Việc phân loại viewpoint dựa vào *mục đích và nội dung*đã cung cấp cho chúng ta một sự hỗ trợ đắc lực cho việc lựa chọn và sử dụng viewpoint sao cho đúng đắng và hợp lý. Không ngoài mục đích trên, tiếp theo, chúng ta được giới thiệu một số những giai đoạn trong quá trình sử dụng viewpoint được đúc kết từ kinh nghiệm thực tế của nhiều kiến trúc sư:

1. *Xác định giới hạn*: Chọn lựa một hoặc nhiều viewpoint phù hợp, chọn những khía cạnh, những layer cần để mô hình hoặc biểu diễn, xác định những ràng buộc đối với phạm vi cần được mô hình.
2. *Xây dựng view*: Tạo nên hoặc lựa chọn nội dung thực sự của viewpoint, tức là tạo nên hoặc lựa chọn những view phù hợp với viewpoint được sử dụng. Điều này gắn liền với việc lựa chọn một thành phần của một mô hình kiến trúc lớn đã được xây dựng để biểu diễn, hoặc tạo hay làm sáng tỏ một thành phần của mô hình kiến trúc.
3. *Đánh giá*: Đánh giá kết quả của việc biểu diễn những view, Xem có phải những stakeholder đã đồng thuận với việc sử dụng những view này là hợp lý để biểu diễn trong ngữ cảnh này không.
4. *Uỷ thác*: Nếu đã đạt được sự đồng thuận giữa những stakeholder chính, thì bước tiếp theo là tạo những sự uỷ nhiệm cho kết quả đạt được.
5. *Thông tin*: Thông tin với những stakeholder khác về kết quả đạt được.

* Thiết kế viewpoint

Tất cả những phần đã nói phía trên là bước chuẩn bị khá đầy đủ để một kiến trúc sư có thể tạo hay lựa chọn cho mình những viewpoint phù hợp. Bây giờ, điều mà chúng ta quan tâm nhất là việc thực sự thiết kế hay lựa chọn cho mình những viewpoint phù hợp, và biểu diễn những view tương ứng như thế nào. Phần tiếp theo sẽ giới thiệu cho chúng ta về những viewpoint cơ bản sử dụng ngôn ngữ ArchiMate đã được thiết kế. Từ đó chúng ta có thể lựa chọn cho mình hướng thiết kế một viewpoint phù hợp hay lựa chọn từ những viewpoint cơ bản để lấy ra những viewpoint cần thiết cho mình.

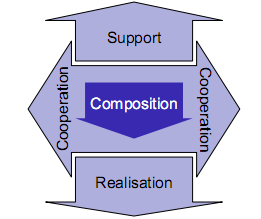
*Thiết kế viewpoint cơ bản*

Như đã phân tích trước, viewpoint là một khái niệm không mới mẻ, được sử dụng cho quá trình phát triển hệ thống thông tin của nhiều tổ chức, mỗi tổ chức có thể sử dụng nhiều ngôn ngữ mô hình kiến trúc khác nhau. Và trong hướng tiếp cận của chúng ta, chúng ta hướng về thiết kế một ArchiMate Viewpoint.

Đối với ngôn ngữ mô hình kiến ArchiMate, dạng viewpoint cơ bản nhất là lựa chọn một tập con phù hợp từ những khái niệm của ArchiMate và biểu diễn những thành phần của kiến trúc bằng những khái niệm này.

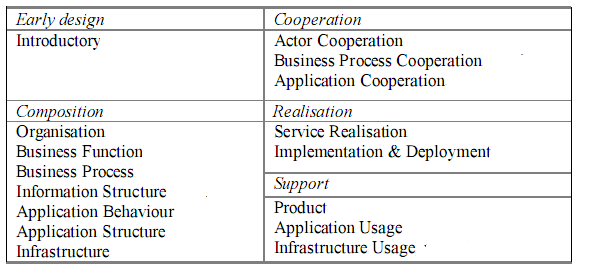
Chúng ta được hướng dẫn để nhận diện những phần tử liên quan đến mô hình của chúng ta:

* *Hướng vào bên trong*: hướng về sự kết hợp bên trong của phần tử
* *Hướng về bên trên*: hướng về những phần tử được hỗ trợ bởi nó
* *Hướng về bên dưới*: hướng về những phần tử hiện thực nó
* *Hướng bên cạnh*: hướng về phần tử cộng tác với nó



1. Hướng dẫn thiết kế viewpoint(ArchiMate D3.4.1a v2.6)

Từ sự kết hợp của những nguyên tắt giúp nhận diện những phần tử liên quan đến mố hình cùng những khái niệm của ArchiMate ở nhiều layer( business, application, technology),nhiều khía cạnh(active,passive,behaviour), và mối quan hệ giữa chúng. Người ta thiết kế ra một tập những viewpoint cơ bản thường được sử dụng với những khái niệm của ngôn ngữ ArchiMate:



1. Các ArchiMate viewpoint cơ bản

Trong những viewpoint cơ bản này, có những viewpoint chỉ được giới hạn ở một layer, hay một khía cạnh, nhưng cũng có những viewpoint thể hiện sự liên kết ở nhiều khía cạnh nhiều layer. Tiếp theo chúng ta sẽ đi vào tìm hiểu chi tiết hơn về những viewpoint này.

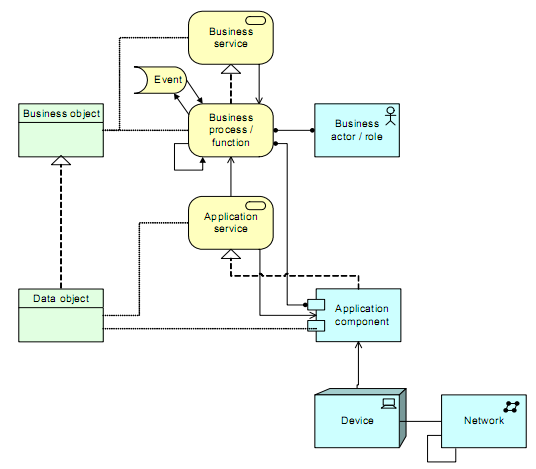
* + 1. **ArchiMate Viewpoint**
* Introductory Viewpoint

Introductory Viewpoint được tạo thành từ một tập con của ngôn ngữ ArchiMate, và được thể hiện bằng những ký hiệu đơn giản. Trong giai đoạn đầu của quá trình kiến trúc, Viewpoint này thường được sử dụng để thiết kế quỹ đạo phát triển của kiến trúc. Lúc này mọi thứ không cần phải được chi tiết. Nó giúp giải thích cơ bản về kiến trúc đối với những người không phải là kiến trúc sư.

Một mục đích khác khi sử dụng viewpoint này đó là việc sử dụng những ký hiệu, những mối quan hệ có định dạnh kém rõ ràng, sẽ giúp những tránh khỏi cảm giác nhầm lẫn đây là kiến trúc đã được thiết kế hoàn tất. Cảm giác này thường rất dễ xảy ra khi sử dụng nhiều cấu trúc chuẩn, chi tiết và trực quan.

|  |  |
| --- | --- |
| Introductory Viewpoint | |
| Stakeholder | kiến trúc sư (Enterprise), nhà quản lý |
| Concern | Lựa chọn ban đầu cho việc thiết kế cấp độ khái quát |
| Mục Đích | Thiết kế, thông tin |
| Cấp độ trừu tượng hoá | Gắn kết, Khái quát |
| Tầng | Nghiệp vụ, ứng dụng, kỹ thuật |
| Khía cạnh | Chủ động, bị động, ứng xử |

Những khái niệm chính được sử dụng trong Introductory Viewpointsẽ được thể hiện trong hình 3-23

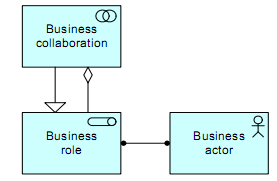


1. Những khái niệm chính được sử dụng trong Introductory Viewpoint

* Khi sử dụng Introductory Viewpoint chúng ta sử dụng những ký hiệu đã được đơn giản và rút gọn đối với những khái niệm, cũng như là những mối quan hệ:
* Tất cả những mối quan hệ ngoại trừ “trigger” và “realization” đều được biểu diễn bằng một đường đơn giản.
* “Trigger” và “realization” đều được thể hiện bằng những mũi tên.
* Bố cục của những view này cũng không ngay ngắn, rõ ràng như trong mô hình kiến trúc ban đầu.
* Organization Viewpoint

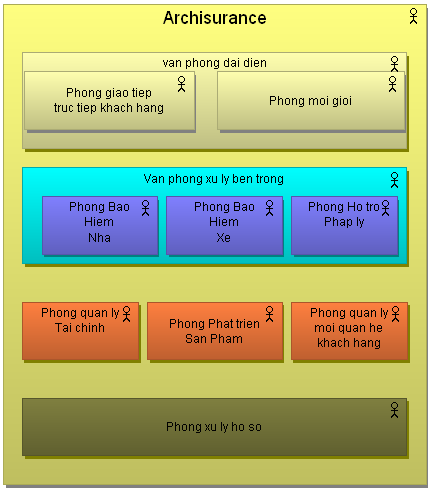
Organization Viewpoint thể hiện kết cấu bên trong một tổ chức của một tập đoàn, hay một phòng ban, một đơn vị thuộc tổ chức, điều đó tuỳ vào việc kiến trúc sư đang đứng ở vị trí nào. Chúng thường được biểu diễn bằng lược đồ gồm những khối lồng vào nhau, ngoài ra, chúng còn có thể được biểu diễn bằng lược đồ “organigram”.

|  |  |
| --- | --- |
| Organization Viewpoint | |
| Stakeholder | Enterprise, process, domain architects, nhà quản lý, nhân viên |
| Concern | Kết cấu bên trong tổ chức, về trách nhiệm và quyền lợi |
| Mục Đích | Thiết kế, thông tin, ra quyết định |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Nghiệp vụ |
| Khía cạnh | Chủ động |



1. Những khái niệm chính được dung trong Organization Viewpoint:

Hình sau là ví dụ về Organization Viewpoint:



1. Ví dụ về Organization Viewpoint

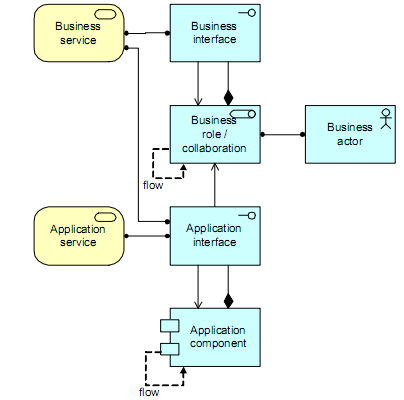
* Actor Cooperation Viewpoint

Actor Cooperation Viewpoint tập trung vào những mối quan hệ giữa những actor với nhau, và giữa actor với môi trường xung quanh nó. Một ví dụ điển hình về điều này là những gì thường được gọi là “lược đồ ngữ cảnh”, chúng đặt một tổ chức vào môi trường, môi trường này gồm có những nhóm tác động bên ngoài như khách hàng, người cung cấp, hay những bộ phận kinh doanh khác. Điều này giúp xác định những sự phụ thuộc bên ngoài, sự cộng tác bên trong của tổ chức.

Một chứa năng quan trọng khác của viewpoint này là thể hiện làm thế nào những actor trong tổ chức cộng tác với nhau để hiện thực những business process, bằng cách thể hiện dòng chảy thông tin giữa chúng.

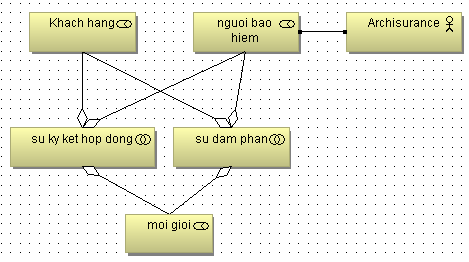
|  |  |
| --- | --- |
| Actor Cooperation Viewpoint | |
| Stakeholder | Enterprise, process, domain architects |
| Concert | Mối quan hệ, và sự cộng tác của những actor với môi trường của chúng |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Cấu trúc, ứng xử |

Những khái niệm chính để xây dựng Actor Cooperation Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Actor Cooperation Viewpoint

Sau đây là ví dụ về Actor Cooperation Viewpoint thể hiện sự cộng tác bên trong tổ chức ArchiSurance:



1. Ví dụ về Actor Cooperation Viewpoint

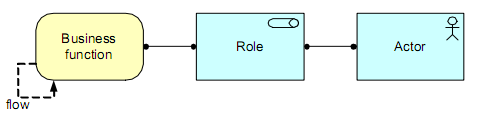
* Business Function Viewpoint

Business Function Viewpoint thường thể hiện những business function chính trong một tổ chức, thể hiện dòng thông tin trao đổi giữa chúng. Business function viewpoint thường được sử dụng để biểu diễn những hoạt động chức năng bền vững của một tổ chức, nó thường không quan tâm đến sự thay đổi của tổ chức, hoặc sự phát triển về công nghệ.

Business Function Viewpoint thường được sử dụng để cung cấp một cái nhìn ở mức trừu tượng hoá về những chức năng tổng quát của một công ty và để cấu trúc những hoạt động chính liên quan đến tổ chức.

|  |  |
| --- | --- |
| Business Function Viewpoint | |
| Stakeholder | Enterprise, process, domain architect |
| Concert | Nhận diện những hoạt động thuộc về bản chất, giảm sự phức tạp. |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Gắn kế |
| Tầng | Nghiệp vụ |
| Khía cạnh | Ứng xử, cấu trúc(chủ động) |

Những khái niệm chính để xây dựng Business Function Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Business Function Viewpoint

Hình dưới là một ví dụ về việc sử dụng Business Function Viewpoint



1. Ví dụ về Business Function Viewpoint

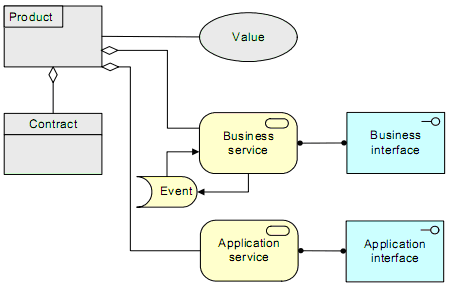
* Product Viewpoint

Product Viewpoint biểu diễn giá trị mà product này cung cấp đến cho khách hàng, hoặc đáp ứng một nhóm những yêu cầu từ bên ngoài, bằng cách tóm lược một tập những dịch vụ, và một contract.

Product Viewpoint thường được dùng trong quá trình phát triển product, chúng dùng để thiết kế product bằng cách kết hợp những những dịch vụ sẵn có, hay nhận diện những dịch vụ mới cần được tạo cho product này, với những giá trị mà khách hàng mong muốn nhận được từ nó. Bên cạnh đó, Product Viewpoint còn có thể được sử dụng như là đầu vào để kiến trúc sư thiết kế một business process.

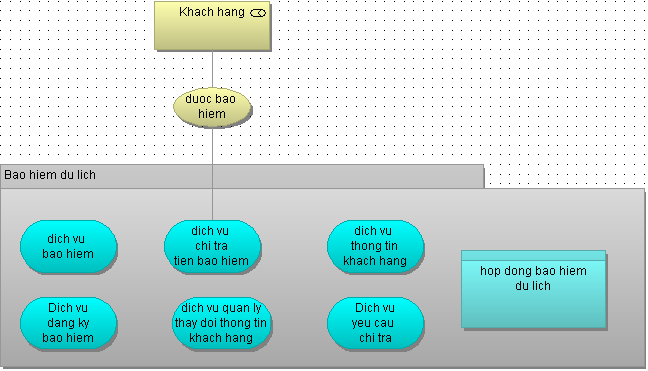
|  |  |
| --- | --- |
| Product Viewpoint | |
| Stakeholder | Người phát triền sản phẩm  Nhà quản lý sản phẩm  Process, domain architect |
| Concert | Sự phát triển sản phẩm  Giá trị được đưa ra bởi những product của một enterprise |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Nghiệp vụ,(ứng dụng) |
| Khía cạnh | Ứng xử, thông tin, cấu trúc(chủ động) |

Những khái niệm chính để xây dựng Product Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Product Viewpoint

Hình dưới là một ví dụ về việc sử dụng Product Viewpoint



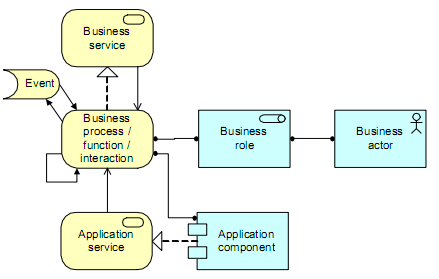
1. Ví dụ về Product Viewpoint

* Service Realisation Viewpoint

Service Realisation Viewpoint được sử dụng để thể hiện một hoặc nhiều business service được hiện thực bởi những business process bên dưới như thế nào. Vì vậy, nó thường là cầu nối giữa Product Viewpoint và Business process Viewpoint. Nó cung cấp cái nhìn từ bên ngoài vào một hay nhiều business process.

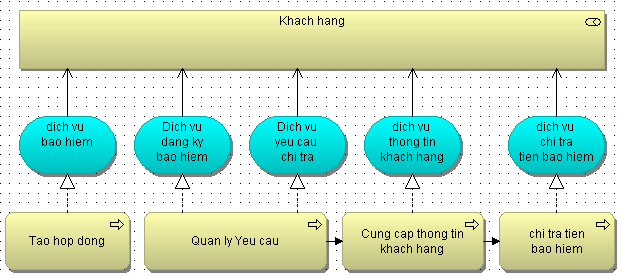
|  |  |
| --- | --- |
| Service Realisation Viewpoint | |
| Stakeholder | Process, domain architect,  người quản lý sản phẩm và chức năng |
| Concert | Giá trị thêm vào của business process về tính nhất quán, tính toàn vẹn, trách nhiệm |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Ứng xử, cấu trúc |

Những khái niệm chính để xây dựng Service Realisation Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Service Realisation Viewpoint

Hình dưới là một ví dụ về việc sử dụng Service Realisation Viewpoint



1. Ví dụ về Service Realisation Viewpoint

* Business Process Cooperation Viewpoint

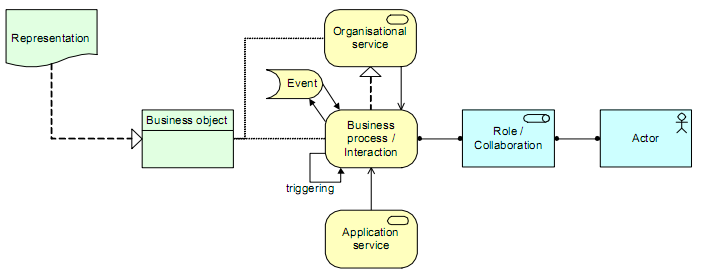
Một Business Process Cooperation Viewpoint thường được sử dụng để thể hiện mối quan hệ của một hay nhiều business process với nhau, cũng như những mối quan hệ xung quanh nó. Viewpoint này vừa có thể được dùng để thiết kế ở mức high-level một business process với ngữ cảnh của nó, vừa có thể hỗ trợ cho nhà quản lý chức năng quản lý những quy trình này bằng cách cung cấp sự thấu hiểu về sự phụ thuộc bên trong của những quy trình này. Những khía cạnh quan trọng được liệt kê sau đây:

* Mối quan hệ về nguyên nhân-kết quả giữa những business process chính của hệ thống.
* Ánh xạ business process vào business function
* Hiện thực những dịch vụ bởi business process
* Việc chia sẻ dữ liệu
* Sự thực thi một business process bởi những vai trò hay những actor giống nhau.

Mỗi một khía cạnh này có thể được thể hiện như là một sub-viewpoint của Business Process Cooperation Viewpoint

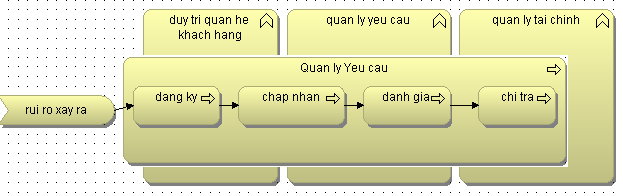
|  |  |
| --- | --- |
| Business Process Cooperation Viewpoint | |
| Stakeholder | Process, domain architects  Nhà quản lý chức năng |
| Những Mối Quan Tâm(concert) | Sự phụ thuộc của những business process, tính toàn vẹn, tính nhất quán, trách nhiệm |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Ứng xử, cấu trúc |

Những khái niệm chính để xây dựng Business Process Collarboration Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Business Process Collarboration Viewpoint

Hình dưới là ví dụ về việc sử dụng Business Process Collarboration Viewpoint



1. Ví dụ về Business Process Collarboration Viewpoint

* Business Process Viewpoint

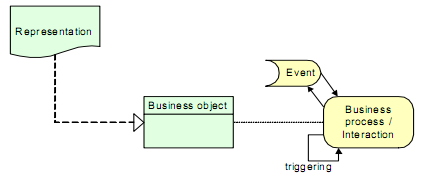
Business Process Viewpoint được sử dụng để thể hiện cấu trúc high-level và sự kết hợp của một hay nhiều business process. Viewpoint này có thể chứa những mối quan hệ trực tiếp giữa các khái niệm như:

* Những dịch vụ mà business process cung cấp ra thế giới bên ngoài, thấy được làm thế nào một quy trình có thể góp phần hiện thực một thành phẩm
* Sự gán kết giữa business role và business process.
* Thông tin được sử dụng bởi business process.

Mỗi một vấn đề trên cũng đều có thể được thể hiện như là một sub-viewpoint của một Business Process Viewpoint.

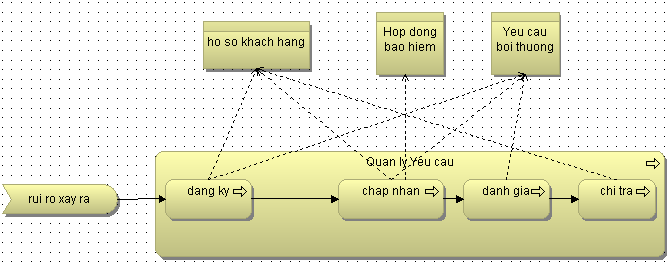
|  |  |
| --- | --- |
| Business Process Viewpoint | |
| Stakeholder | Process, domain architects  Nhà quản lý chức năng |
| Những Mối Quan Tâm(concert) | Cấu trúc của một business process, tính nhất quán, tính toàn vẹn và trách nhiệm |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Ứng xử, Cấu trúc |

Những khái niệm chính thường dùng để xây dựng Business Process Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Business Process Viewpoint

Hình dưới là ví dụ về việc sử dụng Business Process Viewpoint



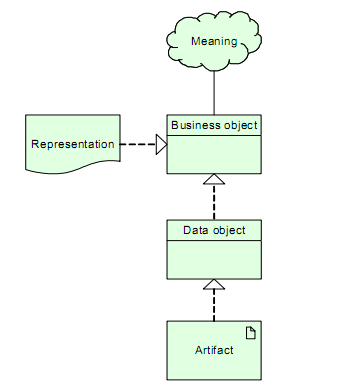
1. Ví dụ về Business Process Viewpoint

* Information Structure Viewpoint

Information Structure Viewpoint là ý tưởng đúng đắn nhất để thể hiện một mô hình thông tintruyền thống, được tạo trong quá trình phát triển của hầu hết mọi hệ thống thông tin. Nó thể hiệu cấu trúc thông tin được sử dụng trong một enterprise hay được sử dụng trong một business process hay application ở dạng kiểu dữ liệu, hay ở dạng cấu trúc những lớp(class-theo hướng đối tượng). Nó còn cho thấy được thông tin ở cấp độ business có thể biểu diễn ở cấp độ application như thế nào thông qua cấu trúc dữ liệu được sử dụng ở đây, và làm thế nào những thông tin này được ánh xạ vào cơ sở hạ tầng bên dưới thông qua cơ sở dữ liệu.

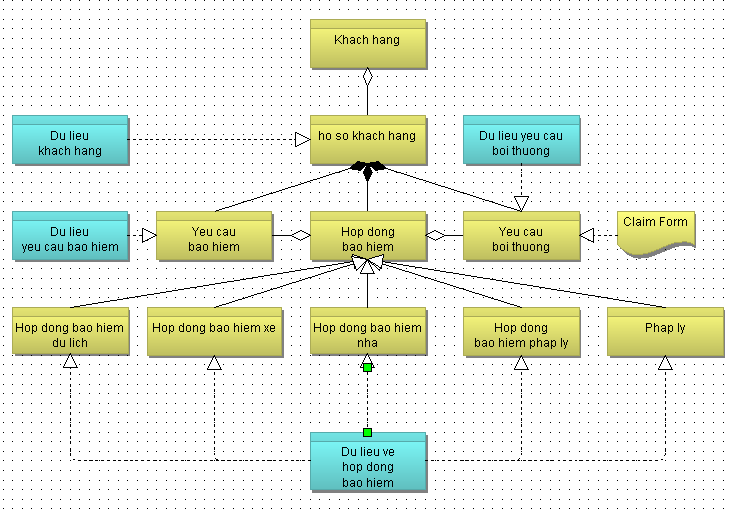
|  |  |
| --- | --- |
| Information Structure Viewpoint | |
| Stakeholder | Domain, information architect |
| Concert | Cấu trúc và sự phụ thuộc của thông tin và dữ liệu được sử dụng. |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Cấu trúc (bị động) |

Những khái niệm chính thường dùng để xây dựng Information Structure Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Information Structure Viewpoint

Hình dưới là ví dụ về việc sử dụng Information Structure Viewpoint



1. Ví dụ về Information Structure Viewpoint

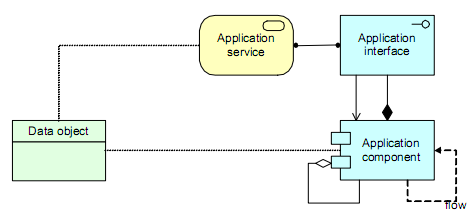
* Application Cooperation Viewpoint

Application Cooperation Viewpoint thể hiện mối quan hệ giữa những ứng dụng hay giữa những thành phần ứng dụng. Viewpoint này mô tả sự phụ thuộc trong luồng trao đổi thông tin giữa những thành phần ứng dụng, hoặc mô tả dịch vụ được đưa ra hay được sử dụng bởi chúng. Viewpoint này thường được sử dụng để tạo một cái nhìn tổng quát về các thành phần ứng dụng trong một tổ chức.

Viewpoint này cũng thường được sử dụng để thể hiện sự điều phối những application service trong việc hỗ trợ hiện thực những business process. Bằng việc mô hình những sự phụ thuộc bên trong của những dịch vụ, sự cộng tác của những ứng dụng bên dưới được thiết lập trong nhiều cách độc lập hơn.

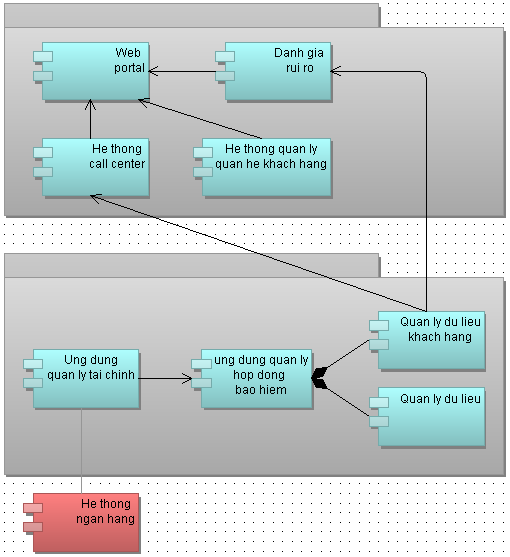
|  |  |
| --- | --- |
| Application Cooperation Viewpoint | |
| Stakeholder | Enterprise, application, domain architect |
| Concern | Mối quan hệ và sự phụ thuộc của những ứng dụng, sự điều phối dịch vụ, giảm sự phức tạp. |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Gắn kết, chi tiết |
| Tầng | Ứng dụng |
| Khía cạnh | Cấu trúc, ứng xử |

Những khái niệm chính thường dùng để xây dựng Application Cooperation Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Application Cooperation Viewpoint

Hình dưới là ví dụ về việc sử dụng Application Cooperation Viewpoint



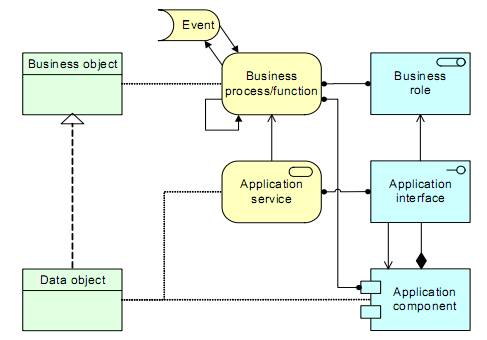
1. Ví dụ về Application Cooperation Viewpoint

* Application Usage Viewpoint

Application usage Viewpoint mô tả cách ứng dụng được sử dụng để hỗ trợ một hoặc nhiều business process, và chúng được sử dụng bởi những ứng dụng khác như thế nào. Viewpoint này có thể được dùng để thiết kế một application bằng cách nhận diện những dịch vụ cần sử dụng bởi business process và những ứng dụng khác, hoặc thiết kế business process bằng cách mô tả những dịch vụ sẵn có.

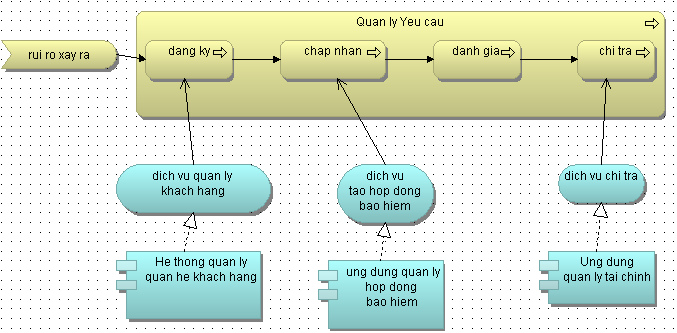
|  |  |
| --- | --- |
| Application usage Viewpoint | |
| Stakeholder | Enterprise, process, application architect, nhà quản lý chức năng |
| Concern | Tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Nghiệp vụ, ứng dụng |
| Khía cạnh | Chủ động, ứng xử, bị động |

Những khái niệm chính thường dùng để xây dựng Application Usage Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Application usage Viewpoint

Hình dưới là ví dụ về việc sử dụng Application Usage Viewpoint



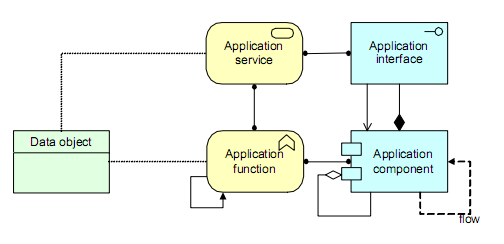
1. Ví dụ về Application usage Viewpoint

* Application Behaviour Viewpoint

Application Behaviour Viewpoint mô tả ứng xử bên trong của một hay của những thành phần ứng dụng, ví dụ, khi nó hiện thực một hay nhiều application service. Viewpoint này thường hữu ích khi khi thiết kế những ứng xử chính của ứng dụng hay thành phần ứng dụng, hoặc trong việc nhận diện những chức năng chồng lên nhau giữa những ứng dụng khác nhau.

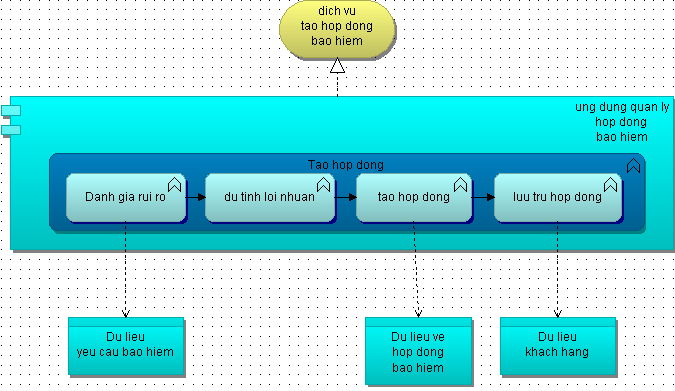
|  |  |
| --- | --- |
| Application Behaviour Viewpoint | |
| Stakeholder | Enterprise, application, domain architect |
| Concern | Cấu trúc,mối quan hệ và sự phụ thuộc của những ứng dụng, tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Ứng dụng |
| Khía cạnh | Chủ động, bị động, ứng xử |

Những khái niệm chính thường dùng để xây dựng Application usage Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong *Application Behaviour Viewpoint*

Hình dưới là ví dụ về việc sử dụng *Application Behaviour Viewpoint*



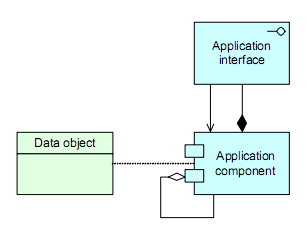
1. Ví dụ về *Application Behaviour Viewpoint*

* Application Structure Viewpoint

Application Structure Viewpoint thể hiện cấu trúc của một hoặc nhiều ứng dụng hay thành phần ứng dụng. Viewpoint này được sử dụng để thiết kế hoặc để có thể hiểu được cấu trúc chính của ứng dụng và dữ liệu kết hợp với chúng. Ví dụ giúp cho bước đầu trong việc phân nhỏ cấu trúc thành từng phần để xây dựng hay nhận diện được những thành phần phù hợp.

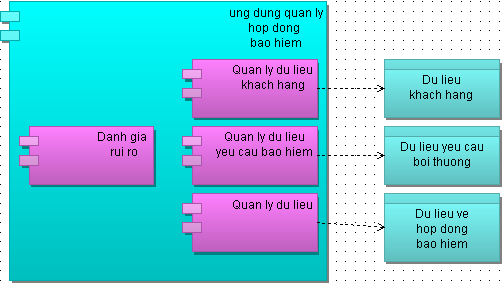
|  |  |
| --- | --- |
| Application Structure Viewpoint | |
| Stakeholder | Enterprise, application, domain architect |
| Concern | Cấu trúc của ứng dụng, tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Ứng dụng |
| Khía cạnh | Chủ động, bị động |

Những khái niệm chính thường dùng để xây dựng Application Structure Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Application Structure Viewpoint

Hình dưới là ví dụ về việc sử dụng Application Structure Viewpoint



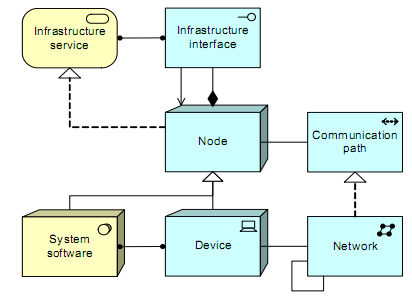
1. Ví dụ về Application Structure Viewpoint

* Infrastructure Viewpoint

Infrastructure Viewpoint bao gồm cơ sở hạ tầng về phần cứng và phần mềm đi theo, những yếu tố mà tầng ứng dụng phụ thuộc vào. Nó chứa các thiết bị vật lý, hệ thống mạng, và phần mềm hỗ trợ như hệ điều hành,cơ sở dữ liệu, phần mềm trung gian.

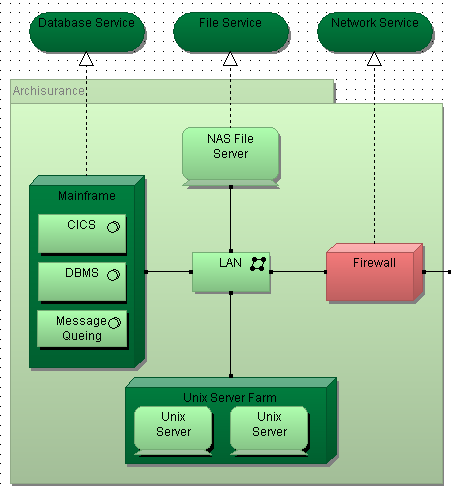
|  |  |
| --- | --- |
| Infrastructure Viewpoint | |
| Stakeholder | Infrastructure architects, (application architects), Nhà quản lý chức năng |
| Concern | Sự ổn định, tính bảo mật, sự ổn định, sự phụ thuộc của cơ sở hạ tầng, chi phí |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Chi tiết |
| Tầng | Kỹ thuật |
| Khía cạnh | Ứng xử, cấu trúc(chủ động) |

Những khái niệm chính thường dùng để xây dựng Infrastructure Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Infrastructure Viewpoint

Hình là ví dụ về việc sử dụng Infrastructure Viewpoint



1. Ví dụ về Infrastructure Viewpoint

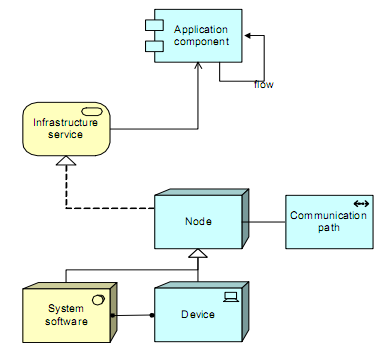
* Infrastructure Usage Viewpoint

Infrastructure Usage viewpoint thể hiện ứng dụng được hỗ trợ bởi cơ sở hạ tầng về phần mềm và phần cứng như thế nào: infrastructure service được đưa ra bởi device, system software, và networks được cung cấp đến ứng dụng.

Viewpoint này đóng vai trò quan trọng trong việc phân tích tính hiệu quả và tính linh động của hệ thống, vì nó thể hiện sự liên quan giữa cơ sở hạ tầng vật lý với ứng dụng luận lý. Nó hữu ích trong việc xác định yêu cầu về tính hiệu quả và chất lượng của cơ sở hạ tầng dựa trên những đòi hỏi đa dạng của những ứng dụng sử dụng nó.

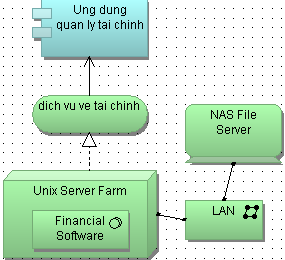
|  |  |
| --- | --- |
| Infrastructure Usage viewpoint | |
| Stakeholder | Application, infrastructure architects  Operational managers |
| Concern | Sự ổn định, tính bảo mật, sự ổn định,sự phụ thuộc của cơ sở hạ tầng |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Ứng dụng, kỹ thuật |
| Khía cạnh | Ứng xử, cấu trúc(chủ động) |

Những khái niệm chính thường dùng để xây dựng Infrastructure Usage viewpoint được thể hiện ở hình bên dưới:



1. Ví dụ về Infrastructure Usage viewpoint

Hình ví dụ về việc sử dụng Infrastructure Usage viewpoint



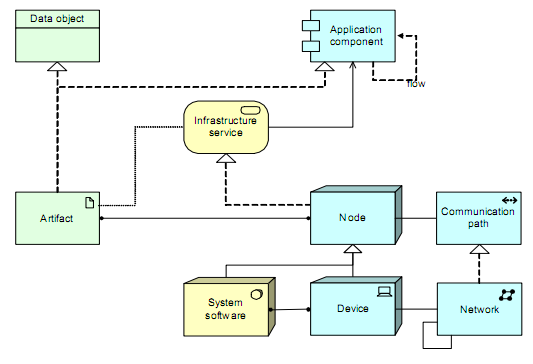
1. Các khái niệm sử dụng trong Infrastructure Usage viewpoint

* Implement và Deloyment Viewpoint

Implement và Deloyment Viewpoint thể hiện cách để một ứng dụng được triển khai trên cơ sở hạ tầng. Bao gồm việc ánh xạ ứng dụng hay thành phần ứng dụng lên trên một artifact vật lý ví dụ như Enterprise Java Bean, và ánh xạ những thông tin được sử dụng bởi những ứng dụng này vào thiết bị lưu trữ ví dụ như những bảng cơ sở dữ liệu, hoặc những tập tin. Về vấn đề bảo mật và phân tích rủi ro, Deloyment viewpoint cũng được sử dụng để nhận diện những mối nguy hiểm và rủi ro.

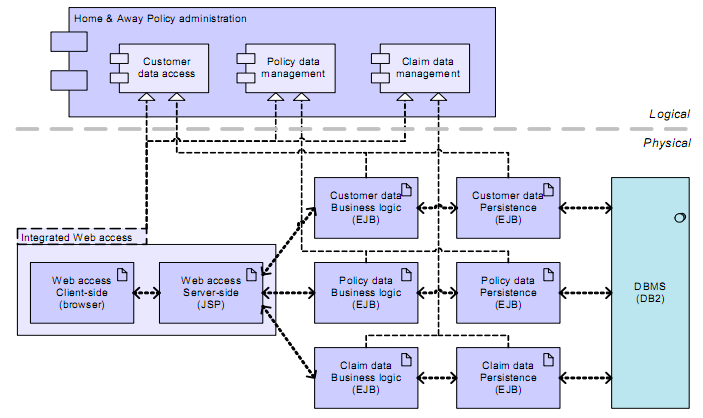
|  |  |
| --- | --- |
| Implement & Deloyment Viewpoint | |
| Stakeholder | Application, infrastructure architect  Giám đốc chức năng |
| Concern | Bảo mật, rủi ro, và các thành phần phụ thuộc |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Gắn kết |
| Tầng | Ứng dụng, kỹ thuật |
| Khía cạnh | Chủ động, ứng xử, bị động |

Những khái niệm chính thường dùng để xây dựng Implement & Deloyment Viewpoint được thể hiện ở hình bên dưới:



1. Các khái niệm sử dụng trong Implement & Deloyment Viewpoint

Hình là ví dụ về việc sử dụng Implement & Deloyment Viewpoint



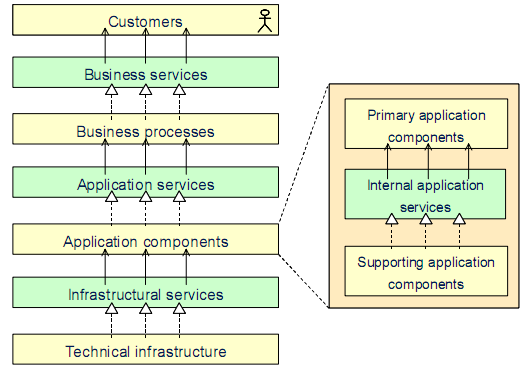
1. Ví dụ về Implement và Deloyment Viewpoint(Enterprise Architecture At Work)

* Layer Viewpoint

Layer Viewpoint thể hiện nhiều layer, nhiều khía cạnh trong một lược đồ, Mục đích chính của viewpoint này là cung cấp một cái nhìn tổng quát về một phần của một kiến trúc enterprise trong một bức tranh. Xa hơn nữa, nó có thể được sử dụng để đánh giá tác động của sự thay đổi hoặc phân tích hiệu quả khi mở rộng kinh doanh với những dịch vụ mới.

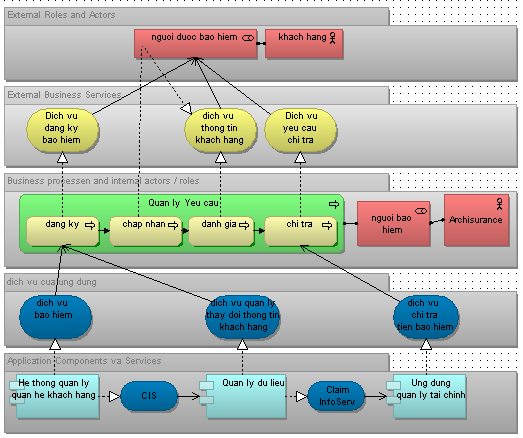
|  |  |
| --- | --- |
| Layer Viewpoint | |
| Stakeholder | Application, infrastructure architect  Giám đốc chức năng |
| Concern | Linh động, giảm độ phức tạp, đánh giá tác động của sự thay đổi, nhất quán |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Khái quát |
| Tầng | Nghiệp vụ, ứng dụng, kỹ thuật |
| Khía cạnh | Cấu trúc, ứng xử, thông tin |

Lược đồ dưới sẽ thể hiện cấu trúc tổng quát của các Layer Viewpoint



1. Cấu trúc Layer Viewpoint

Hình ví dụ về việc sử dụng Layer Viewpoint



1. Ví dụ về Layer Viewpoint
   * 1. **Viewpoint Framework**

Như đã được trình bày ở những phần trước, mục đích chính cho sự tồn tại của những framework về viewpoint là để giúp những kiến trúc sư thoát khỏi gánh nặng của việc phải chọn được những viewpoint phù hợp với ngữ cảnh kiến trúc hiện tại. Hiện nay, đã có một số những viewpoint được phát triển mạnh mẽ và phổ biến như: *Zachman framework (Zachman 1987), Kruchten’s 4+1 view model (Kruchten 1995), RM-ODP (ITU 1996), and TOGAF (The Open Group 2002).* Và thực tế bên cạnh đó, nhiều tổ chức lớn cũng đã định nghĩa cho riêng họ những framework về viewpoint, để có thể mô tả kiến trúc họ một cách hiệu quả hơn. Trong phần này chúng ta sẽ được giới thiệu một cách tổng quát về hai framework: “The 4+1 View Model” và “RM-ODP”.

* The 4+1 View Model

1995 Kruchen giới thiệu một framework về viewpoint bao gồm năm viewpoint. Những viewpoint này cho phép phân biệt những mối quan tâm của những stakeholder khác nhau của một kiến trúc: người dùng cuối, kỹ sư phát triển phần mềm, kỹ sư hệ thống, người quản lý dự án,...cũng như là phân biệt yêu cầu chức năng và phi chức năng. Chi tiết về framework “The 4+1 View Model” sẽ được trình bày qua bảng sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Viewpoint** | Logical | Process | Development | Physical | Scenarios |
| **Goal** | Nắm bắt những service mà hệ thống phải cung cấp | Nắm bắt những khía cạnh đồng thời, và khía cạnh đồng bộ của thiết kế | Mô tả khía cạnh thuộc về tổ chức của phần mềm, và sự phát triển của nó | Mô tả việc ánh xạ một phần mềm trên phần cứng, cũng như sự phân bố giữa chúng | Cung cấp động lực để khám phá những phần tử trong việc thiết kế đánh giá và minh hoạ |
| **Stakeholder** | Architect, end-user | Architectect, System- designer, Intergrator | Architect  Developer  Manager | Architect  System-designer | Architect  end-user  Developer |
| **Concerns** | Thuộc về chức năng | Hiệu quả, tính sẵn sàng, đánh giá khuyết điểm… | Tính linh động, khả năng tái sử dụng, kết cấu tổ chức… | Tính hiệu quả, tính sẵn sáng, và khả năng mở rộng… | Tính sinh động, dễ hiểu |
| **Meta-model** | Object, Class, Associations, Inheritance  … | Event, Message, Broadcast  … | Module  Subsystem  Layer  … | Processor  Device  Bandwidth  … | Objects  Events  Steps |

* RM-ODP

Năm 1996 The Reference Model for Open Distributed Processing(RM-ODP) được giới thiệu như là kết quả của nỗ lực kết hợp của hai tổ chức chuẩn hoá quốc tế ISO và ITU để phát triển một framework điều phối cho vấn đề chuẩn hoá những quy trình mở. Framework này định nghĩa ra năm viewpoint: *enterprise, information, computation, engineering,* và *technology.* Chi tiết về framework RM-ODP sẽ được trình bày trong bảng sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Viewpoint** | Enterprise | Information | Computation | Engineering | Technology |
| **Goal** | Nắm bắt mục đích, phạm vi, chính sách phát triển hệ thống | Nắm bắt thông tin và quy trình được thực hiện bởi hệ thống | Mô tả sự phân loại những đối tượng tương tác trong hệ thống | Mô tả thiết kế những khía cạnh hướng phân tán của hệ thống | Mô tả sự lựa chọn những công nghệ được sử dụng trong hệ thống |
| **Concern** | Yêu cầu của tổ chức, và cấu trúc | Sự thông tin, yêu cầu về quy trình | Phân phối chức năng của hệ thống từ sự phân tích chức năng | Sự phân tán của hệ thống, và những chức năng, những kỹ thuật cần thiết | Sự lựa chọn phần cứng, phần mềm… |
| **Meta-model** | Objects  Communi-  ties  Permissions  Obligations  Contract… | Object classes  Associations  Process  … | Objects  Interfaces  Interaction  Activities  ... | Objects  Channels  Node  Capsule  Cluster  ... | Không rõ ràng |

* + 1. **Archimate và UML**

Trong thực tế, ta có thể hình dung, ArchiMate mô tả cấu trúc của một thành phố, trong khi UML mô tả cấu trúc của những ngôi nhà và cao ốc văn phòng, cả hai đều là cần thiết, và chúng giải quyết vấn đề khác nhau. Cả hai ngôn ngữ phục vụ cho mục đích khác nhau và có các bên liên quan khác nhau.

UML thường được sử dụng bởi kỹ sư thiết kế hệ thống, cái mà UML còn thiếu là một metamodel nhất quán và thuần nhất để có thể định nghĩa những thành phần của những mô hình khác nhau quan hệ với nhau như thế nào. Nói cách khác, UML không trả lời câu hỏi: dịch vụ IT và quy trình nghiệp vụ quan hệ với nhau như thế nào. Trong khi, ArchiMate có làm được điều đó. ArchiMate bắt đầu với việc nắm rõ những mối quan hệ phức tạp giữa quy trình nghiệp vụ và dịch vụ IT bên trong những miền khác nhau từ động lực kinh doanh cho đến quy trình nghiệp vụ, quản lý dịch vụ cho đến hệ thống và cơ sở hạ tầng. ArchiMate định nghĩa một metamodel cho phép những mối quan hệ này được cấu trúc, được ràng buộc và truyền thông với nhau, trong khi đó UML không bị ràng buộc giữa những loại mô hình .

ArchiMate có sử dụng lại những hình ảnh ký hiệu tuơng tự như UML nhưng định nghĩa lại ngữ nghĩa của chúng. Archimate thường được sử dụng để mô hình kiến trúc một enterprise còn UML thường mô tả sự thực thi hệ thống từ những viewpoint khác nhau.

* 1. **SOA**

SOA – viết tắc của cụm từ Service Oriented Architect là một trong những kiến trúc hệ thống định nghĩa việc phát triển và tích hợp cách thành phần chức năng xoay quanh các quy trình nghiệp vụ và đóng gói các thành phần đó thành những gói tương thích với nhau.

* + 1. **Giới thiệu tổng quan**

SOA là cách tiếp cận để xây dựng hệ thống CNTT cho phép doanh nghiệp tận dụng những gì đang có và dễ dàng thay đổi theo yêu cầu để hỗ trợ doanh nghiệp. Có thể hiểu, khi xây dựng hệ thống ứng dụng, quy trình của doanh nghiệp được mô hình hóa và từng thành phần trong quy trình được xem như một nguồn cung cấp cho ứng dụng khác hoặc ngược lại. Kiến trúc SOA không tạo ra một hệ thống thông tin phân cấp mà là trực tiếp, xuyên suốt, có thể thay đổi quy trình xử lý và tăng khả năng khai thác thông tin toàn diện hơn.

Do dựa vào quy trình và xem những chức năng là 1 khối nên nhu cầu phát sinh như cấu trúc lại doanh nghiệp, cung cấp dịch vụ mới có thể được tái tạo lại từ hệ thống dễ dàng. Hơn nữa, dựa trên ngôn ngữ XML (dùng để truyền tải dữ liệu) nên những dịch vụ trong kiến trúc SOA không phụ thuộc vào nền tảng và có thể sử dụng lại cho những nhu cầu mới phát sinh.

Mặc dù SOA đem lại cho doanh nghiệp nhiều lợi ích, lợi thế cạnh tranh từ hệ thống thông tin nhưng cũng đòi hỏi doanh nghiệp phải có chiến lược đầu tư, phương pháp và công cụ để hỗ trợ xây dựng kiến trúc.

* + 1. **Lịch sử SOA**
* **1994**- Alexander Pasik: đưa ra thuật ngữ SOA để nhấn mạnh cụm từ “server orientation” trong quan niệm Client/Servier.
* **1996**- Gartner analysts, Roy W. Schulte: Web Services không phải là SOA và ngược lại
* **2000**- Các dịch vụ web của Microsoft làm cho SOA trở thành xu thế chủ đạo.
* **Từ 2000 trở đi** : Các nhà cung cấp khác cũng tham gia như: IBM, Oracle, HP, SAP, and SUN. (B2B hype)
  + 1. **Tại sao dùng SOA**

Trong thực tế CNTT thì trong toàn bộ hệ thống các ứng dụng, phần mềm hệ thống, cơ sở hạ tầng… thường không đồng nhất. Một số ứng dụng được sử dụng để chạy các quy trình kinh doanh hiện tại do đó bắt đầu xây dựng một cơ sở hạ tầng mới không phải là một lựa chọn.

Các doanh nghiệp cần nhanh chống đáp ứng với sự thay đổi kinh doanh thật nhanh nhẹn. Đầu tư tận dụng các ứng dụng và cơ sở hạ tầng hiện có để giải quyết nhu cầu kinh doanh mới hơn là sự lựa chọn khôn ngoan. SOA cho phép doanh nghiệp có thể thêm các ứng dụng mới hoặc nâng cấp các ứng dụng hiện tại để đáp ứng các nhu cầu nghiệp vụ mới, cung cấp tùy biến cho các lựa chọn khác nhau giúp bảo vệ cơ sở hạ tầng hiện có, tiết kiệm kinh phí cho doanh nghiệp.

* + 1. **SOA giải quyết 4 bài toán**
* Tối ưu quy trình.
* Cung cấp dịch vụ mới .
* Khai thác thông tin tổng thể.
* Quản lý hạ tầng thông tin ở kiến trúc mới.
  + 1. **Vậy SOA là gì ?**

SOA - Service Oriented Architecture (Kiến trúc Định hướng Dịch vụ), là 'Khái niệm về hệ thống trong đó mỗi ứng dụng được xem như một nguồn cung cấp dịch vụ'.

Một Service-Oriented Architecture là một phương pháp để chia sẻ các function của một ứng dụng một cách rộng rải và mềm dẻo.

Từ góc nhìn của doanh nghiệp, có thể thấy tinh thần của SOA được thể hiện thông qua 4 khái niệm cơ bản là :

* “Dịch vụ” (service).
* “Quy trình nghiệp vụ” (business process).
* “Ứng dụng tổ hợp” (composite application).
* Quản lý quy trình nghiệp vụ (Business Process management - BPM).
  + 1. **SOA không phải là một kỹ thuật**

Không nên nhìn SOA như một kiến trúc kỹ thuật vì SOA không có định nghĩa kỹ thuật nào chi tiết. Nếu nói về kỹ thuật, một gói dịch vụ EJB phải tương tác được với một component.NET. Như thế SOA không định ra chuẩn gì về kỹ thuật mà người lập trình cần quan tâm. SOA hoàn toàn phải được tiếp cận và thiết kế từ góc độ người quản lý - những người chịu trách nhiệm mô hình hoá nghiệp vụ của doanh nghiệp thành những nghiệp vụ có thể thực hiện bằng IT.

* + 1. **Dịch vụ trong SOA**

Dịch vụ là yếu tố then chốt trong SOA. Có thể hiểu dịch vụ như là hàm chức năng (mô-đun phần mềm) thực hiện qui trình nghiệp vụ nào đó. Một cách cơ bản, SOA là tập hợp các dịch vụ kết nối 'mềm dẻo' với nhau (nghĩa là một ứng dụng có thể 'nói chuyện' với một ứng dụng khác mà không cần biết các chi tiết kỹ thuật bên trong), có giao tiếp (dùng để gọi hàm dịch vụ) được định nghĩa rõ ràng và độc lập với nền tảng hệ thống, và có thể tái sử dụng. SOA là cấp độ cao hơn của phát triển ứng dụng, chú trọng đến qui trình nghiệp vụ và dùng giao tiếp chuẩn để giúp che đi sự phức tạp kỹ thuật bên dưới.

* + 1. **Định nghĩa “Dịch vụ” trong SOA**

Một dịch vụ là một hàm chức năng cho phép một hoặc nhiều yêu cầu gọi đến và trả về một hoặc nhiều kết quả thông qua một giao tiếp chuẩn.

Các dịch vụ có thể xử lý riêng biệt từng phần của một công việc chẳng hạn như sửa hoặc xử lý một transaction.

Các dịch vụ thì không phụ thuộc vào trạng thái của các dịch vụ khác.

* + 1. **SOA tách riêng thực hiện thực dịch vụ với giao tiếp gọi dịch vụ.**

Thiết kế SOA tách riêng phần thực hiện dịch vụ (phần mềm) với giao tiếp gọi dịch vụ. Điều này tạo nên một giao tiếp nhất quán cho ứng dụng khách (client) sử dụng dịch vụ bất chấp 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 có thể triển khai và tái sử dụng trong toàn bộ quy trình nghiệp vụ. Điều này cho phép tái sử dụng phần mềm tốt hơn, cũng như tăng sự linh hoạt vì nhà phát triển có thể cải tiến dịch vụ mà không làm ảnh hưởng đến ứng dụng client sử dụng dịch vụ.

* + 1. **Ưu điểm**

Ưu điểm quan trọng nhất của SOA là khả năng kết nối 'mềm dẻo' (nhờ sự chuẩn hóa giao tiếp) và tái sử dụng. Các dịch vụ có thể được sử dụng với trình client chạy 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 web .NET và ngược lại).

* + 1. **Hai nguyên tắc thiết kế SOA**
* **Mô-đun**: Tách vấn đề lớn thành nhiều vấn đề nhỏ.
* **Đóng gói**: Che đi dữ liệu và lô-gic trong từng mô-dun (hay 'hộp đen') đối với truy cập từ ngoài
  + 1. **Triển khai SOA cho doanh nghiệp**
* SOA có thể phát triển trên các công nghệ hiện có

Doanh nghiệp sẽ quan tâm tới chi phí phát triển hay chuyển đổi hệ thống sang SOA. Bởi vì SOA không định nghĩa kỹ thuật nên về bản chất, chi phí không nằm ở chỗ phải đào tạo nhân sự hay tìm hiểu kỹ thuật mới.

Trên thực tế, trừ khi doanh nghiệp lớn và sử dụng các gói dịch vụ trọn gói cho doanh nghiệp của IBM, Oracle hay SAP, nếu doanh nghiệp sử dụng các phần mềm từ nhiều supplier khác nhau, việc nâng cấp hay quyết định chọn phần mềm nào chủ yếu phải đặt trên tiêu chuẩn cơ bản của SOA - sự tương thích theo chuẩn. Phát triển hệ thống SOA có nghĩa là đưa các phần mềm hiện có từ chỗ hoàn thành yêu cầu nghiệp vụ của doanh nghiệp độc lập tới chỗ hoàn thành và đưa ra kết quả là những dữ liệu có khả năng sử dụng bởi phần mềm khác, và ở mức cao hơn, cho phép 2 phần mềm sử dụng trực tiếp các dịch vụ của nhau.

* SOA phải được xây dựng từ từ

Không có doanh nghiệp nào đứng yên hay thiết lập cứng nghiệp vụ kinh doanh của mình. Vì lý do đó, bản chất SOA là mô hình hoá nghiệp vụ kinh doanh cũng sẽ phải liên tục thay đổi và cập nhật. Một giải pháp phát triển SOA kéo dài một năm trở lên là không bao giờ phù hợp. Nói một cách khác, nếu một dự án làm SOA cho một phần của hệ thống tuyên bố rằng một năm sau, hệ thống của doanh nghiệp sẽ làm được những điều A, B, C thì đó là một ước mơ không có thực.

 Ngược lại SOA cần triển khai nhanh từng thành phần nhỏ và luôn phải để tâm đến **chuẩn**. Đây là vấn đề khó với doanh nghiệp việt nam bởi nhân sự không được đào tạo bài bản. Nói một cách đơn giản, chúng ta có nhiều thợ, có thể sửa chữa những thứ đã được làm ra nhưng lại có ít kiến trúc sư có thể làm ra những cái mà người khác sử được. Có vài kiến trúc sư làm ra những cái mà... chính họ sửa được.

* + 1. **Giai đoạn định nghĩa SOA**

SOA không được định nghĩa trước mà luôn phải được định nghĩa vào lúc phát triển hệ thống cho doanh nghiệp cụ thể.

Từ 6 điểm nói trên, có thể tóm lại 4 thành phần cần định nghĩa khi phát triển một hệ thống SOA cho doanh nghiệp là:

* Quy trình nghiệp vụ của doanh nghiệp, tại mỗi quy trình đó đầu vào là gì, kết quả dữ liệu là gì và đâu là những dữ liệu cần được chia sẻ giữa các quy trình với nhau.
* Ngữ cảnh sử dụng nghiệp vụ và mức độ ưu tiên trong đó những dịch vụ cơ bản phải được làm trước.
* Hạ tầng IT về mạng, server, thiết bị lưu trữ, phân quyền các phòng ban,...
* Kế hoạch phát triển và đưa vào sử dụng các phần mềm trong đó quan tâm nhiều về vấn đề công nghệ nền tảng và chi phí triển khai.
  1. **Open ESB**

Một bản thiết kế thì cần có công cụ và kỹ thuật để hiện thực nó và SOA cũng vậy. Kiến trúc SOA cũng cần phải có công cụ và kỹ thuật để hiện thực.

OpenESB chính là công cụ giúp ta hiện thực kiến trúc SOA.

* + 1. **Giới thiệu**

OpenESB là một dự án mã nguồn mở với mục đích xây dựng một Enterprise Service Bus chuẩn quốc tế. Có địa chỉ website là: *https://open-esb.dev.java.net*

Một ESB - Enterprise Service Bus cung cấp sự linh hoạt và dễ mở rộng cho việc xây dựng ứng dụng SOA và các giải pháp tích hợp ứng dụng. IBM - “ ESB được đặt vào hệ thống CNTT của doanh nghiệp, có thể xem như doanh nghiệp đó đó đang ứng dụng SOA.

OpenESB sử dụng Netbeans IDE và Glassfish Application Server. Tích hợp chúng thầy một bộ công cụ gọi là GlassFish ESB.

* + 1. **GlassFish ESB**

GlassFish ESB là mội gói phân phối bởi OpenESB chứa đựng runtime và một số thành phần thiết yếu.

Chỉ cần tải GlassFish ESB là bạn có mọi thứ để phát triển, cài đặt và chạy các ứng dụng tích hợp được hổ trợ bằng giao diện đồ họa.

Nó đã được thử nghiệm để đáp ứng các tiêu chí phát hành nghiêm ngặt liên quan đến hiệu suất và chất lượng.

* + 1. **Tại sao dùng ESB để hiện thực SOA**

Mục đích của OpenESB là nhằm giúp cho việc hội nhập và phát triển SOA hiệu quả hơn bằng cách cung cấp các thành phần hổ trợ “out of the box”. Ví dụ như việc quét một thư mục, với sự hổ trợ của OpenESB bạn không cần phải code một đoạn code nào.

BPEL SE là một thành phần “out of the box”. Với thành phần này, business logic được thể hiện trong WS-BPEL, một ngôn ngữ rất ý nghĩa giúp các logic phức tạp được đơn giản hóa bởi vài dòng code hoặc một hình họa.

OpenESB cũng giúp giải quyết các khó khăn thường gặp và các pattern, như xử lý lỗi, patten tích hợp …

Xuyên suốt OpenESB, việc cấu hình thì được ưa thích hơn coding và kết quả là phát triển ứng dụng nhanh hơn và code ít hơn.

* + 1. **Các thành phần “out of the box” mà OpenESB hổ trợ.**

OpenESB hổ trợ rất nhiều các thành phần “Component” chia thành các mục chính.

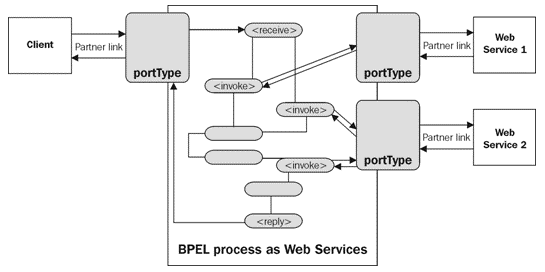
* Logic and orchestration: [*BPEL SE*](https://open-esb.dev.java.net/BPELSE.html)*,* [*Camel SE*](http://wiki.open-esb.java.net/Wiki.jsp?page=CamelSE)*,* [*IEP SE*](https://open-esb.dev.java.net/IEPSE.html)*,* [*Java EE SE*](http://wiki.open-esb.java.net/Wiki.jsp?page=JavaEESE) …
* Basic interfacing: [*eMail BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=EmailBC)*,* [*File BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=FileBC)*,* [*FTP BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=FTPBC)*,* [*HTTP BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=HTTPBC) …
* Databases and data manipulation: [*Database BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=DatabaseBC)*,* [*Data Integrator / ETL SE*](http://wiki.open-esb.java.net/Wiki.jsp?page=ETLSE) …
* Other interfaces: [*Asterisk BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=AsteriskBC)*,* [*Exec BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=ExecBC)*,* [*HL7 BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=HL7BC)*,* [*SNMP BC*](https://open-esb.dev.java.net/SNMPBC.html) …
* EIS interfaces: [*Jbi4Cics (CICS BC)*](https://open-esb.dev.java.net/CICSBC.html)*,* [*CORBA BC*](http://wiki.open-esb.java.net/Wiki.jsp?page=CORBABC)*,* [*DCOM BC*](https://open-esb.dev.java.net/DCOMBC.html)*,* [*EJBBC*](http://wiki.open-esb.java.net/Wiki.jsp?page=EJBBC)…
* Other domains: [*X3D SE*](http://wiki.open-esb.java.net/Wiki.jsp?page=X3DSE)
* No longer maintained: [*ADABAS Natural BC*](https://open-esb.dev.java.net/ADABASNatural.html)*,* [*AOSD*](https://open-esb.dev.java.net/AOSD.html)*,* [*SMTP BC*](https://open-esb.dev.java.net/SMTPBC.html)…

Và còn rất nhiều thành phần khác có thể xem tại *https://open-esb.dev.java.net/Components.html*

* + 1. **BPEL - Business Process Execution Language**

Trong kiến trúc SOA, Business process là một thành phần quan trọng. Có những quy trình hết sức phức tạp, nên việc có một ngôn ngũ chuyên đặc tả các quy trình là một điều hết sức cần thiết để rút ngắn thời gian hiện thực, logic quy trình dễ hiểu, dễ hiện thực.

BPEL là một ngôn ngữ mạnh dùng để trợ giúp phát triển các ứng dụng phức tạp, lớn, gồm nhiều thành phần và dịch vụ Web khác. BPEL cho phép bạn mô tả một quy trình dài bằng cách sử dụng trình soạn thảo đồ họa để biểu thị luồng công việc trên các biểu đồ thân thiện với con người.

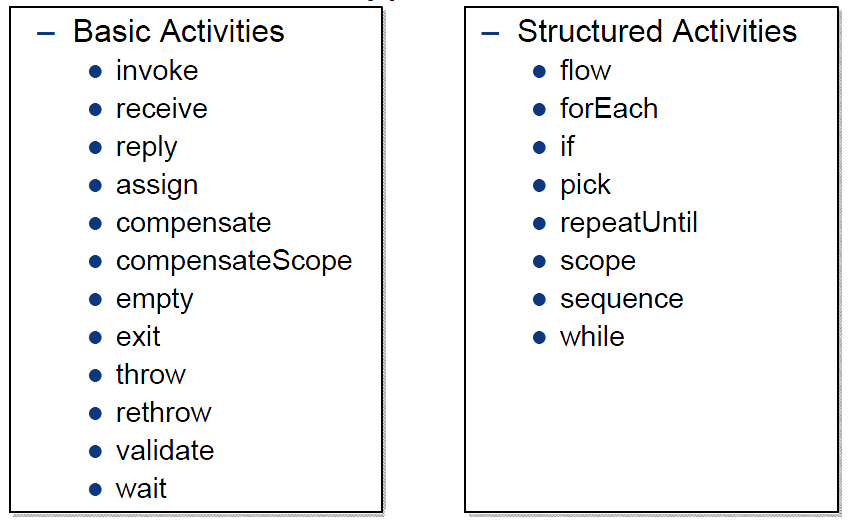


1. Ví dụ một BPEL process
   * 1. **Ta có thể làm gì với BPEL**

BPEL là ngôn nghĩ định nghĩa “business processes” để kết hợp dịch vụ và tạo ra 1 dịch vụ mới. BPEL là ngôn ngữ dựa XML, xây dựng trên WSDL, XML Schema và XPath.

Để định nghĩa 1 “business process collaboration” , các activity phải được định nghĩa, và message trao đổi với các dịch vụ liên quan cũng phải chỉ ra. WSDL cung cấp miêu tả kĩ thuật cơ bản và đặc ta cho việc trao đổi message nhưng lại không đi chi tiết vào tích phối hay tương tác. Vì thế BPEL sẽ định nghĩa các khía cạnh không có trong WSDL.

BPEL cung cấp các câu trúc như vòng lặp, nhánh, biến , gán… Khá giống các ngôn ngữ lập trình truyền thông và cho phép định nghĩa business processes theo cách 1 thuật toán. Các cấu trúc này làm việc định nghĩa process trở nên đơn giản hơn.



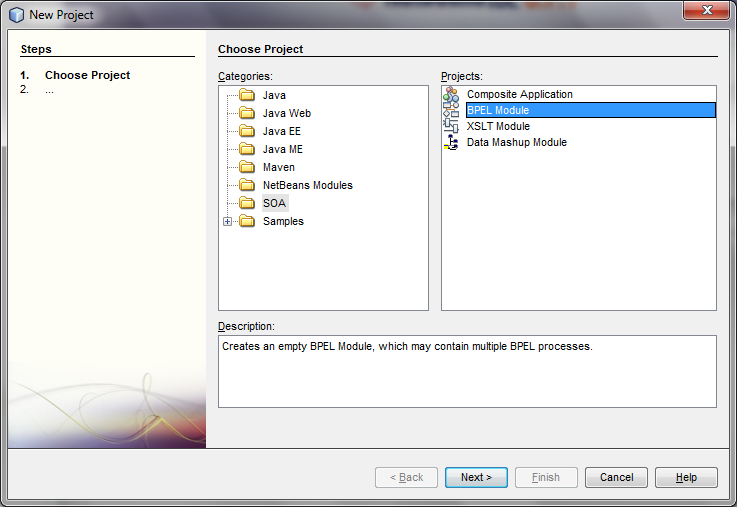
1. Các activities trong BPEL
   * 1. **Cài đặt**

* JDK6 update 21.
* GlassfishESB v2.2. download tại *https://open-esb.dev.java.net/Downloads.html*
  + 1. **HelloWord project**
* Mục đích của demo:
* Cho bạn thấy làm thế nào để tạo một BPEL project
* Làm thế nào để test và deploy BPEL project
* Giải thích những thành phần cơ bản của một BPEL project
* Kịch bảng quy trình

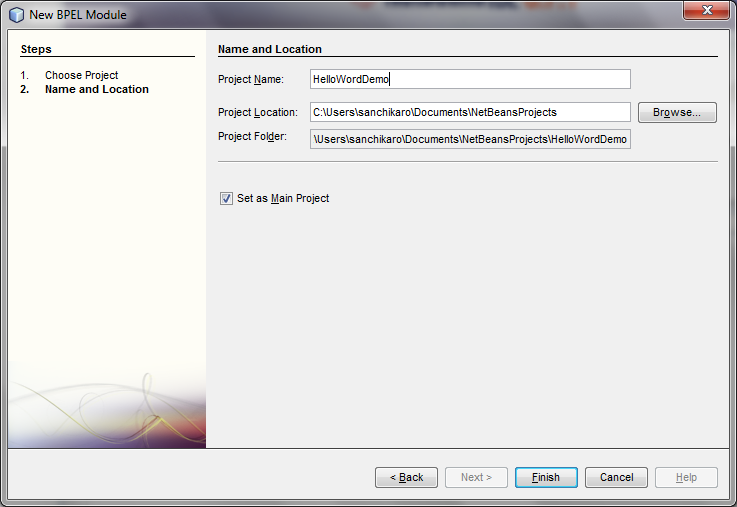
Input đầu vào là tên của người dùng và thông điệp trả về là cụm từ Hello + tên người dùng

* Tạo BPEL project

Tạo BPEL project File/New project

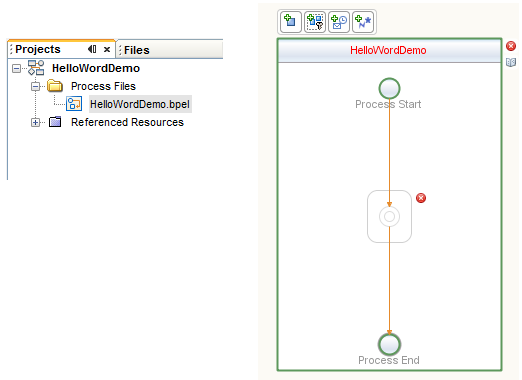


Chọn BPEL Module rồi nhấn Next, cửa sổ tạo BPEL module hiện ra.



Đặt tên cho project, nhấn Finish để hoàn tất.

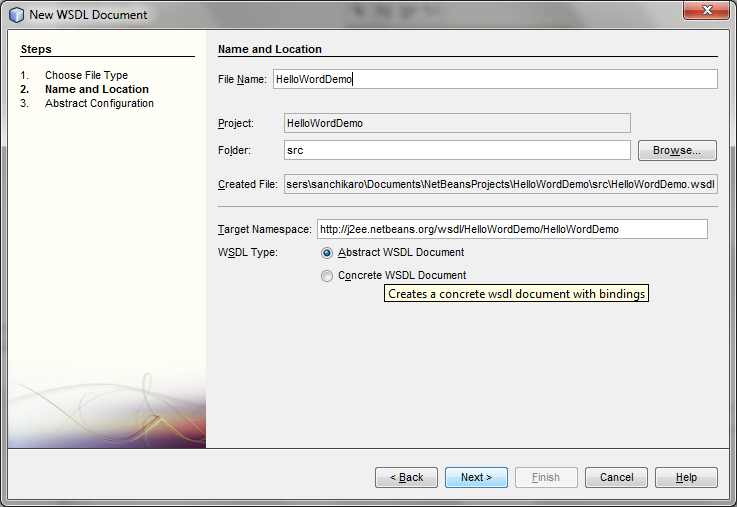
Trong project sẽ tạo sẵng file HelloWordDemo.bpel . File này là file đặc tả quy trình (Business process). Ban đầu quy trình rổng chưa có gì hết như hình bên dưới



* Tạo parlner link

Parlner link là một file WSDL , đại diện cho một Business process, đặt tả các thông tin: Port Type name, Operation name, Operation Type, Biến Input, Biến Output.

Đưa chuột lên biểu tượng của project, nhấn chuột phải chọn New/WSDL Document

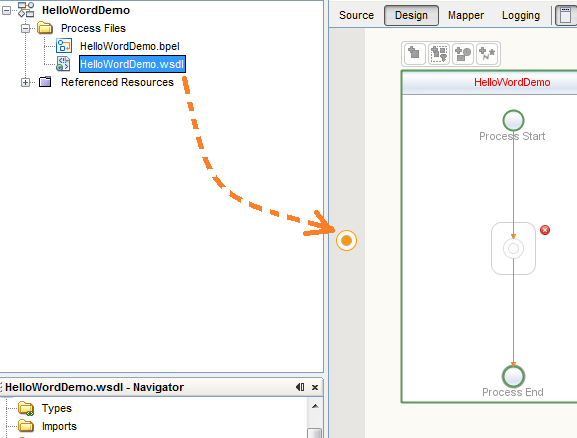


Nhấn chọn Next.

Trong cửa sổ kế tiếp bạn cấu hình tên biến cho input và output, nhấn Finish.



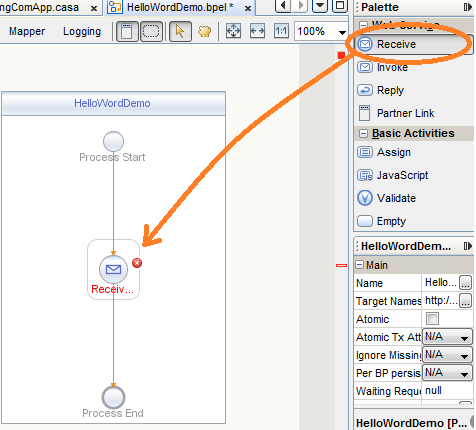
Kéo và thả file HelloWordDemo.wsdl vào partner lane (cột bên trái như hình dưới)



* Cấu hình BPEL Receive

Bắt đầu một quy trình đều phải có một BPEL Receive đại diện cho input message của quy trình.

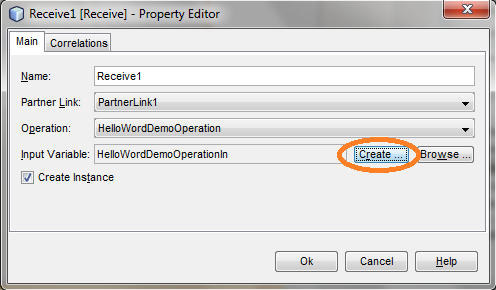
Kéo biểu tượng BPEL Receive từ bảng mẩu vào bên trong BPEL process.



Nhấn vào nút edit đế cấu hình cho BPEL Receive



Trong cửa sổ cấu hình BPEL Receive chúng ta cấu hình tên của BPEL Receive tại khung Name, PartnerLink, tên biến input như hình dưới.

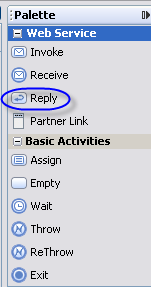


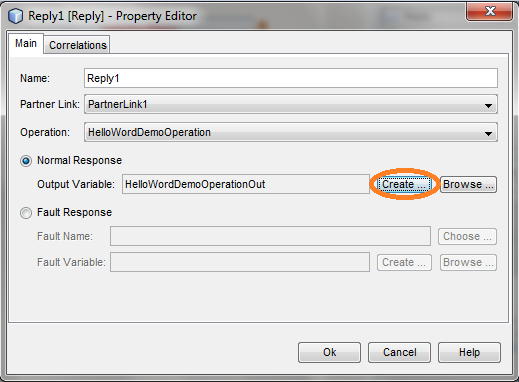
Sau khi cấu hình xong sẽ thấy một mũi tên trỏ từ PartnerLink tới BPEL Receive như hình bên dưới.



* Cấu hình BPEL Reply

Quy trình kết thúc khi gặp một BPEL Reply. Một BPEL Reply đại diện cho output message của quy trình. Cấu hình tương tự như BPEL Receive.



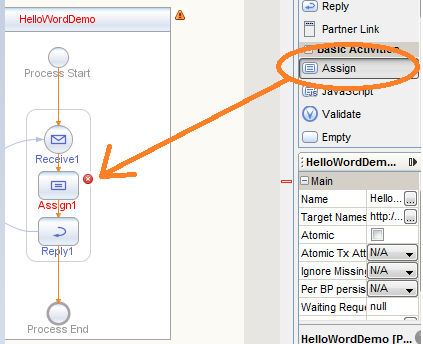




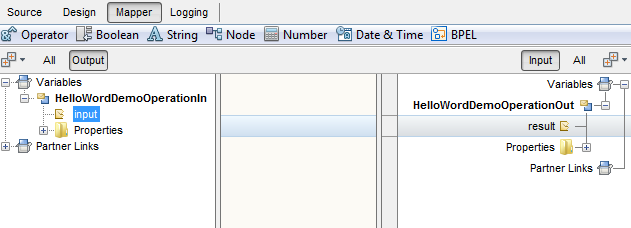
* Assign

Assing là một basic active dùng đề map dữ liệu giữa các biến trong quy trình.

Kéo biểu tượng assign vào trong quy trình như hình vẽ bên dưới.

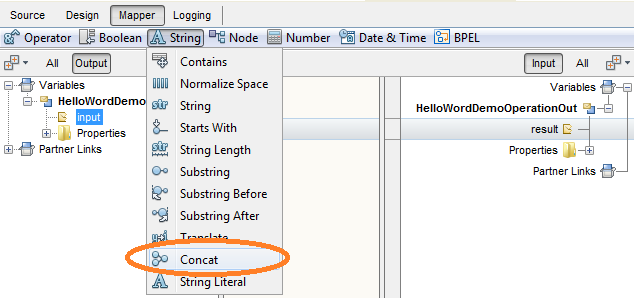


Sau khi kéo assign vào trong process, double click chuột trái vào biểu tượng assign. Giao diện sẽ chuyển sang giao diện Mapping như hình dưới.

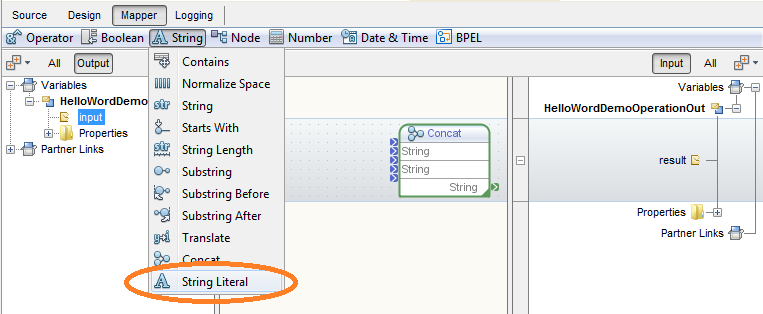


Trong quá trình map dữ liệu chúng ta có thể xử lý biến bằng các function có sẵn và được chia theo nhóm. Ở đây mình sẽ nối chữ “Hello” với biến input nhận vào và gán giá trị đã xử lý cho biến output.

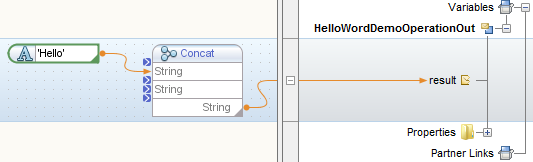
Nhấn chuột vào HelloWordDemoOperationOut/result . Trên thanh function chọn String/Concat.

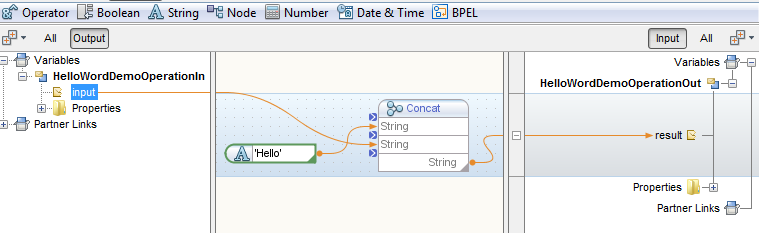


Cũng trên thanh function chọn Sting/String Literal để tạo một chuỗi có giá trị là “Hello”



Kéo chuột để map dữ liệu như hình vẽ





* Tạo Composite Application

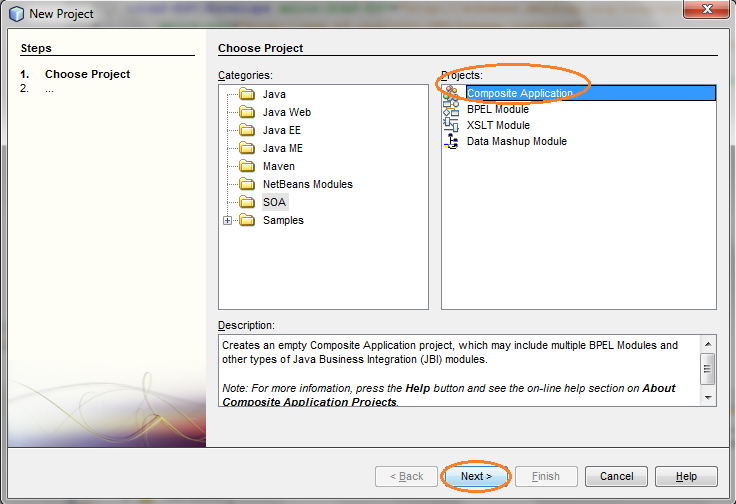
Bạn có thể xem BPEL project giống như là bản vẽ còn Composite Application project chính là chiếc máy hiện thực bản vẻ đó. Nghĩa của cụm từ Composite Application có nghĩa là “Ứng dụng tích hợp. Trên nó mình có thể tích hợp nhiều JBI module (BPEL module là 1 loại JBI module).

Composite App có hai thành phần chính:

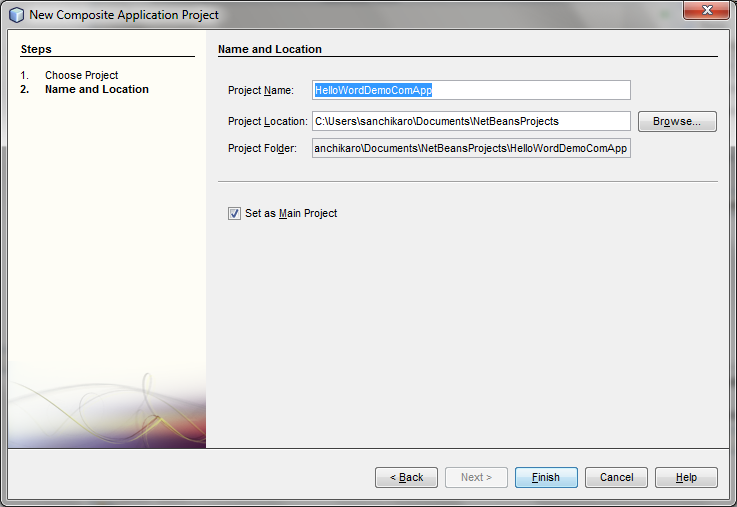
* JBI module: chứa các JBI module như BPEL module, EJB module …
* WSDL port: Cổng kết nối các service ra thế giới bên ngoài. Có thể là các giao thức SOAP, HTTP, FTP, File …

Vào menu chọn New/New Project.

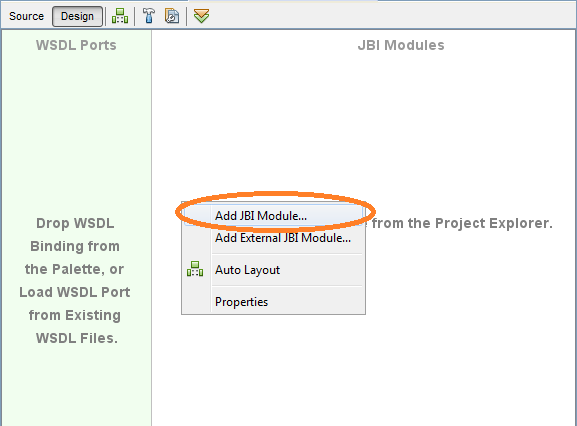
Cửa sổ New project hiện ra chọn SOA/Composite Application nhấn Next.



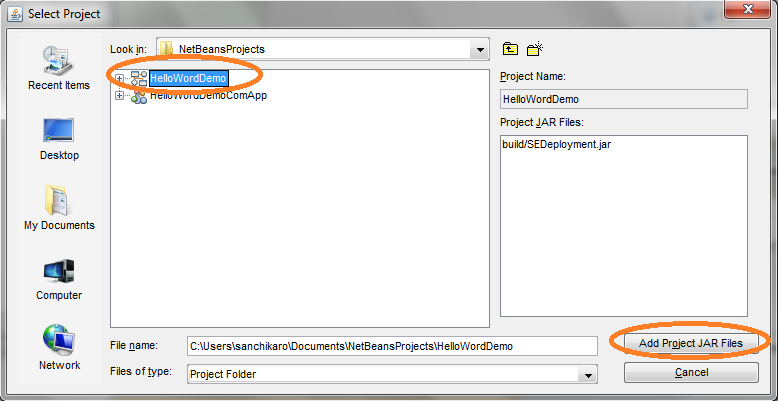
Cửa sổ tạo Composite Application hiện ra bạn gỏ tên project rồi nhấn Finish.



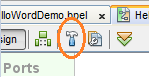
Trong cửa sổ chính bạn nhấn chuột phải chọn Add JBI Module để thêm BPEL module vừa tạo ở trên vào.



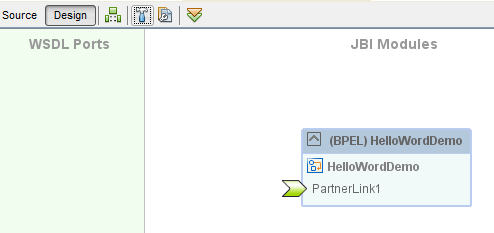
Cửa sổ Select Project hiện ra. Chọn BPLE project HelloWordDemo nhấn Add Project JAR Files.



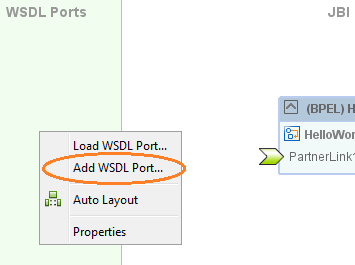
Sau khi add JBI module ta tiến hành build Composite Application bằng cách nhấn vào nút build được khoanh tròn ở hình dưới.



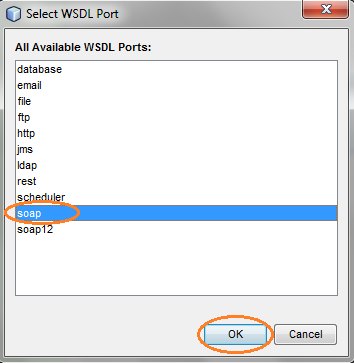
Sau khi build sơ đồ sẽ hiện ra như thế này.



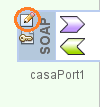
Tiếp đến add WSDL port. Nhấn chuột phải lên cột WSDL port chọn Add WSDL port.



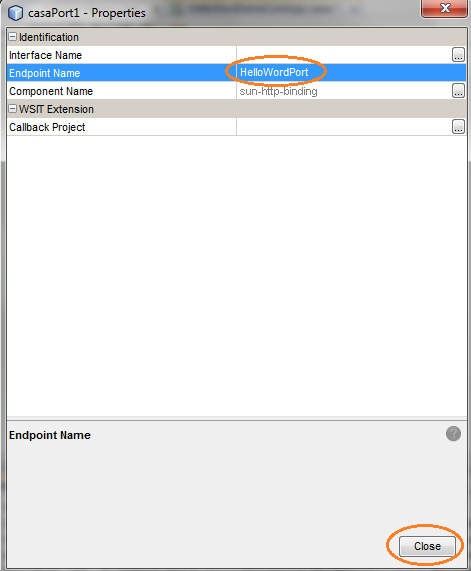
Cửa sổ select WSDL ports hiện ra chọn SOAP . Có nghĩa giao thức để gọi service HelloWordDemo là giao thức SOAP.



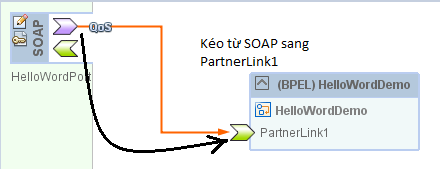
Nhấn edit trên biểu tượng của WSDL port như hình dưới.



Đặt tên cho port.

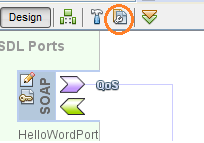


Sau đó dùng chuột kéo 1 đầu từ WSDL port sang PartnerLink của HelloWordDemo service.



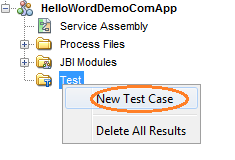
* Deploy

Nhấn build 1 lần nữa rồi nhấn tiếp biểu tượng deploy kế bên để deploy project lên server.

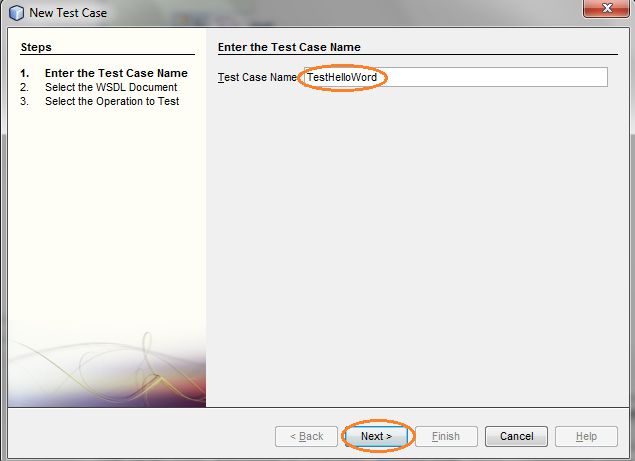


* Test

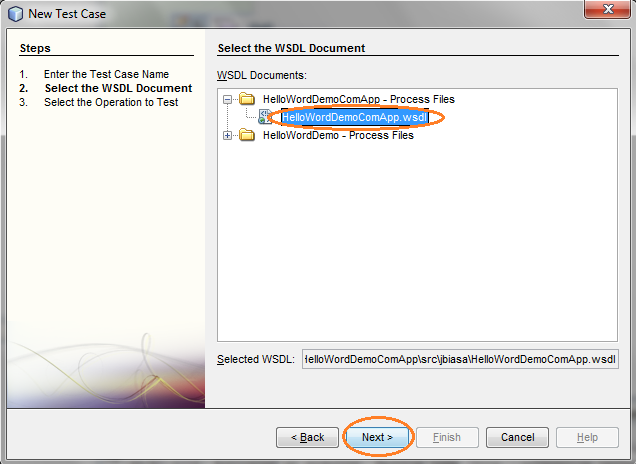
Nhấn chuột phải vào Test của project chọn New Test Case.



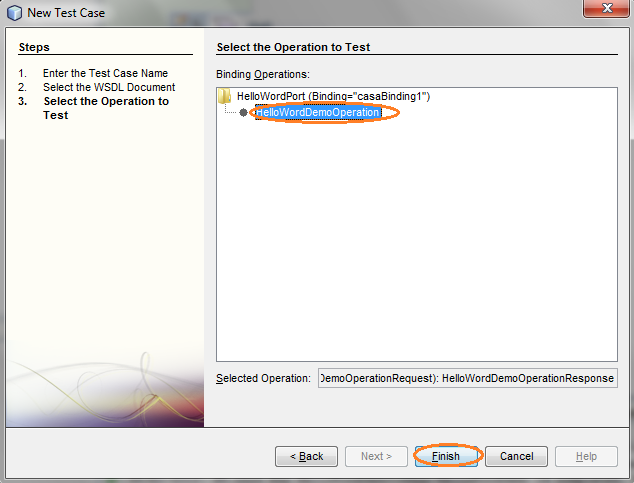
Cửa sổ tao Test case hiện ra gỏ tên của Test case vào nhấn Next.



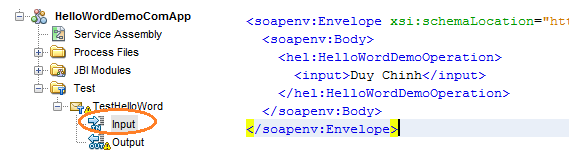
Chọn HelloWordDemoComApp – Process Files/HelloWordDemoComApp.wsdl nhấn Next.



Chọn HelloWordPort/HelloWordDemoOperation nhấn Finish.

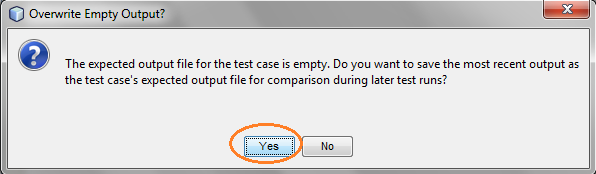


Sau khi xong một TestCase sẽ xuất hiện. Nhấn chuột vào input để hiện đoạn mã XML nhận input. Bạn nhập tên của mình vào cập thẻ <input></input> . Lưu ý “input” chính là tên biến lúc bạn gán khi tạo file WSDL partnerlink.

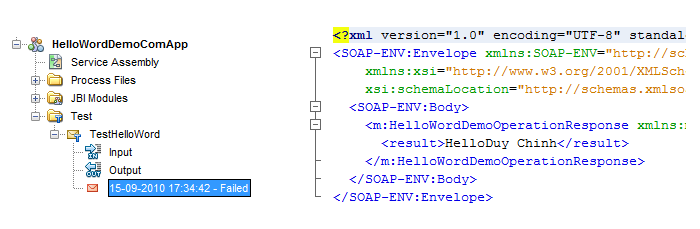


Nhấn chuột phải lên test case chọn Run để thực hiện việc test.

Một cửa sổ hiện ra bạn chọn Yes.



Một message được tạo ra với nội dung là kết quả trả về như hình dưới.



* Tổng kết

Trong ví dụ trên trình bày cho các bạn một số kiến thức cơ bản về BPEL cách tạo một BPEL project như thế nào,cách map dữ liệu giữa các biến, cách deploy là test một BPEL module.

* + 1. **Kết nối database với** [**Database BC**](http://wiki.open-esb.java.net/Wiki.jsp?page=DatabaseBC)

[Database BC](http://wiki.open-esb.java.net/Wiki.jsp?page=DatabaseBC) là một component do OpenESB cung cấp. Giúp đọc và ghi dữ liệu tới database thông qua JDBC.

Trong demo này sẽ hướng dẫn bạn cách chèn dữ liệu xuống một bảng bằng [Database BC](http://wiki.open-esb.java.net/Wiki.jsp?page=DatabaseBC). Chú ý bạn phải tạo sẵn database có tên là helloword.

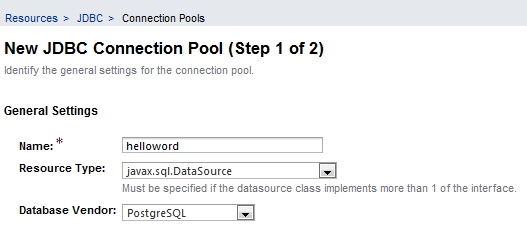
* Cấu hình kết nối database trên Glassfish server.
* Các kết nối xuống database từ BPEL module không thực hiện trực tiếp mà phải qua trung gian là Glassfish server dựa trên một JNDI name.
* Glassfish server sẽ tạo sẵn một connection pool xuống database và quản lý các kết nối tránh hiện tượng quá nhiều kết nối xuống database cùng một lúc.

Vào giao diện quản lý của glassfish server <http://localhost:4848/login.jsf>

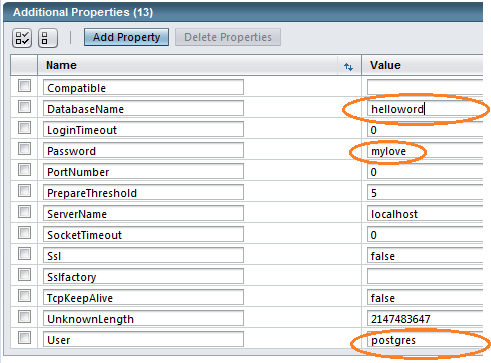
**Tạo connection pool**

Tại giao diện chính, cột bên trái là cây chỉ mục. Di chuyển đến *Resources/JDBC/ Connection Pools* chọn *New* để tạo một connection pool.

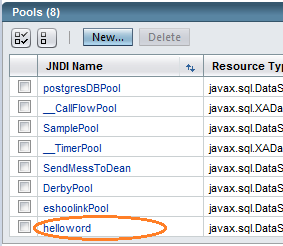
Giao diện khởi tạo hiện ra bạn điền thông tin như hình dưới. Name là tên của connection, Database Vendor chọn PostgreSQL do database sử dụng trong demo này là Postgres8.4



Sau khi nhấn Next qua bước kết tiếp bạn sẽ thiết đặc các thông số kết nối database. Ba thông số quan trọng là : DatabaseName, Password và User.



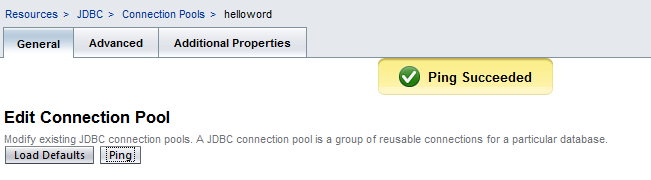
Sau khi nhấn Finish bạn sẽ thấy trong danh sách Pools có helloword vừa được tạo. Click chuột để test kết nối.



Giao diện mới hiện ra, Click vào nút Ping để test kết nối có thành công hay không



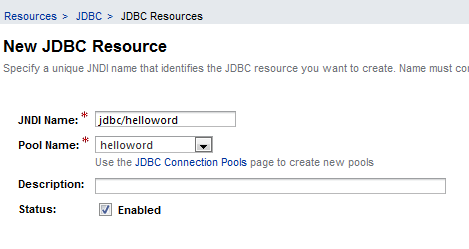
Nếu xuất hiện thông báo như hình dưới thì có nghĩa là đã kết nối thành công. Ngược lại thì thông thường lỗi là do cấu hình thông số không đúng hoặc thiếu driver giành cho postgres. Trong trường hợp thiếu driver thì bạn hãy tìm file postgresql-8.4-701.jdbc3.jar chép vào lib của glassfish và khởi động lại server.



**Tạo JDBC Resource**

JDBC Resource bao gồm một JNDI name máp với một Connection pool. Khi một Bingding muốn kết nối xuống database nó sẽ kết nối nhờ JNDI name.

Vào mục *Resources/JDBC/JDBC Resources* . Giao diện quản lý các JDBC Resources hiện ra. Chọn New để tạo JDBC Resource mới.



JDNI Name bạn điền là *jdbc/helloword*. Pool Name chọn *helloword*. Xong nhấn OK

* Tạo BPEL module

Các kỹ thuật cơ bản đã demo trong ví dụ trước nên sẽ không trình bày lại nữa. Trong phần này sẽ có các kỹ thuật mới sau:

* Tạo một function trong postgres
* Tạo database pingding WSDL
* Gọi service
* XML shema

**Tạo function trên database postgres**

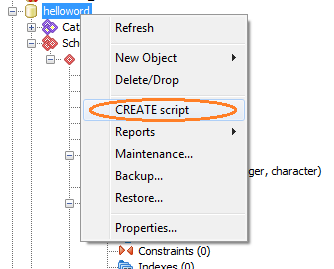
Đối với các loại câu query insert hay update thì ta viết function trực tiếp trên database.

Trong demo này chúng ta có một bảng dữ liệu tên là tb\_users. Chúng ta sẽ viết 1 function insert dữ liệu xuống tb\_users đồng thời trả về id của record vừa insert xuống.

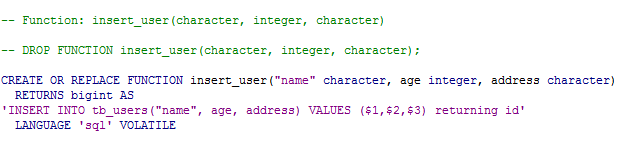


1. Bảng tb\_users

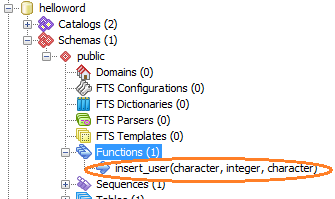
Click chuột phải vào biểu tượng database helloword chọn CREATE script.



Cửa sổ viết script hiện ra bạn hãy gõ đoạn mã này vào



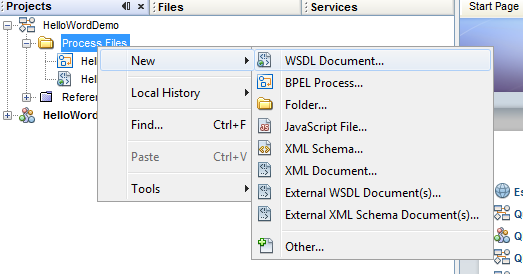
Nhấn nút tam giác  để chạy script.



**Tạo file WSDL database binding**

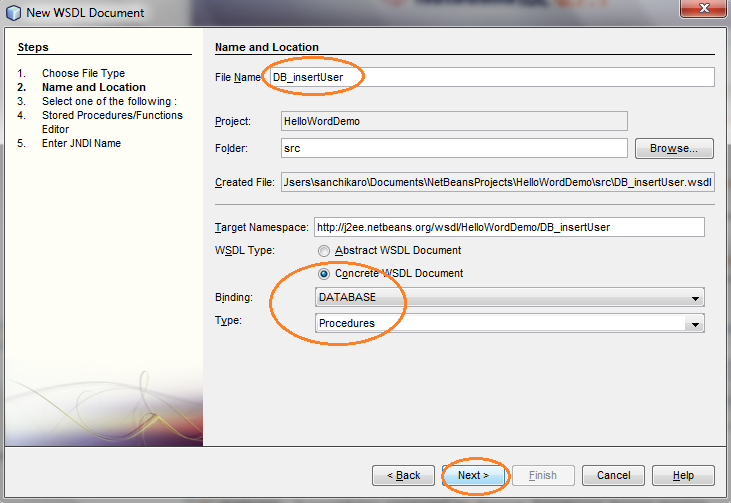
Chúng ta sẽ tạo một file WSDL binding có tên là **DB\_insertUser** .

Click chuột phải vào project chọn New/WSDL Document

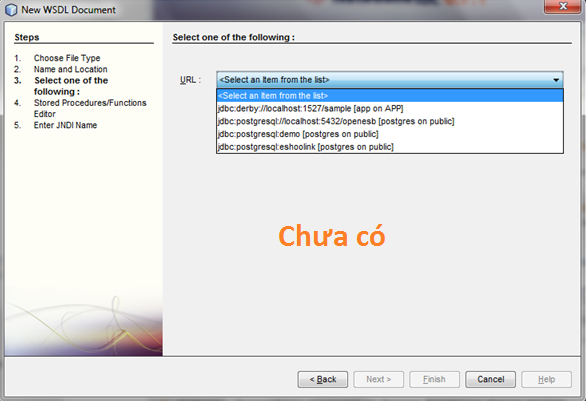
****

Cửa sổ tạo WSDL hiện ra. Bạn cấu hình các thông tin như sau:

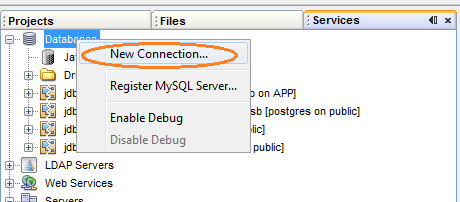
* File name: *DB\_insertUser*
* WSDL Type: *Concrete*
* Binding: *DATABASE*
* Type: *Procedures*



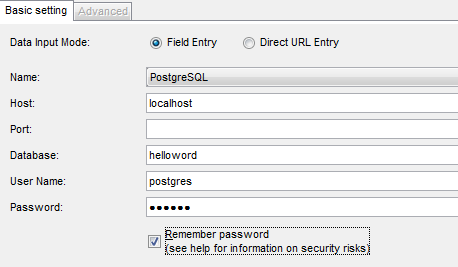
Sau khi nhấn Next bạn sẽ chuyển sang bước 2 là chọn kết nối tới database mình muốn. Trong trường hợp này chúng ta chưa tạo kết nối từ IDE tới database.



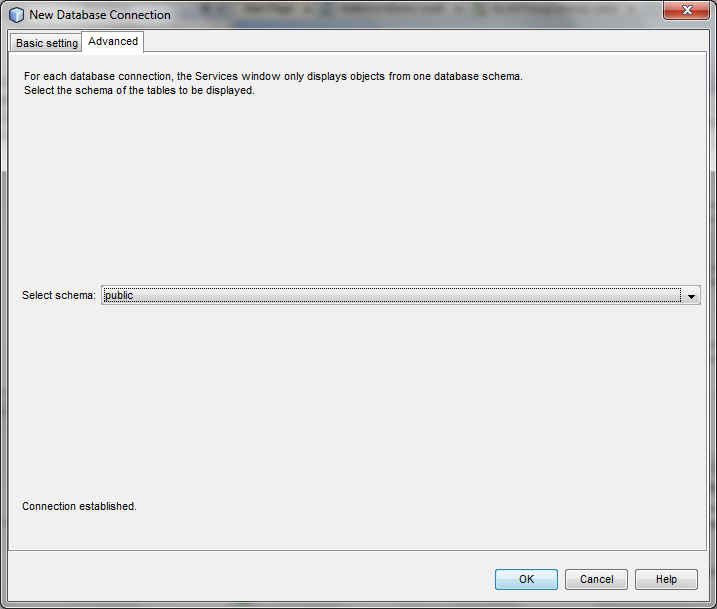
Thoát khổi cửa sổ tạo WSDL. Chúng ta tạo kết nối tới database cho IDE. Chọn tab Services click chuột phải lên Database chọn New Connection



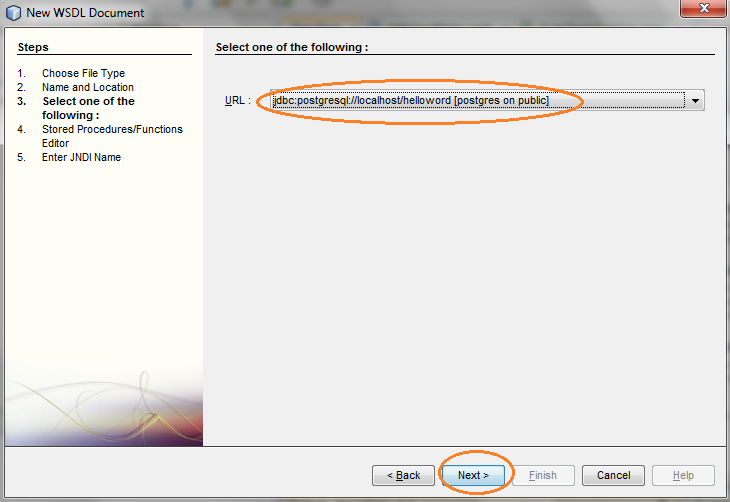
Cửa sổ New Database Connection hiện ra, bạn cấu hình các thông số cần thiết như hình bên dưới rồi nhấn Ok.



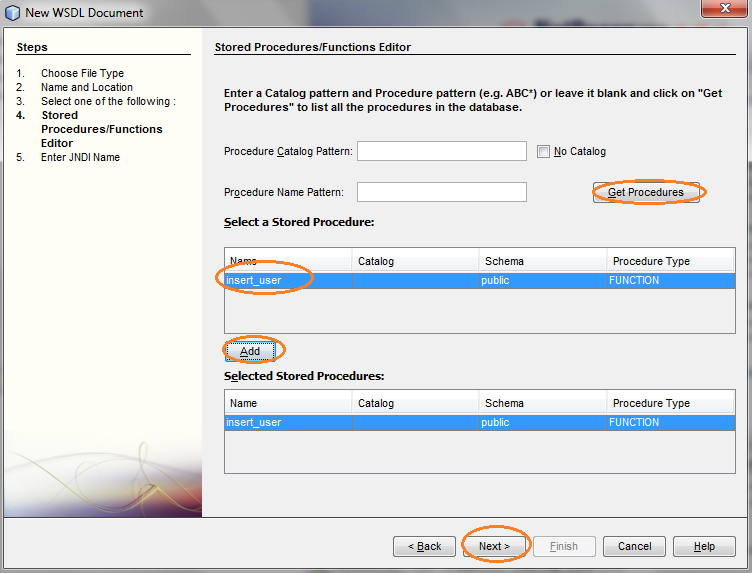
Sau đó nó sẽ yêu cầu bạn chọn schema bạn chọn **public** rồi nhấn OK.

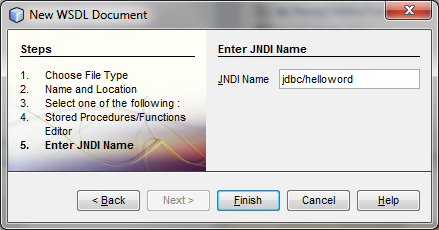


Sau đó quay lại tiếp tục tạo file WSDL binding .



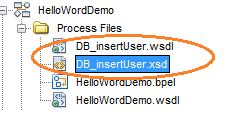
Lúc này bạn đã có URL connect toi database cua mình. Chọn nó rồi nhấn Next.





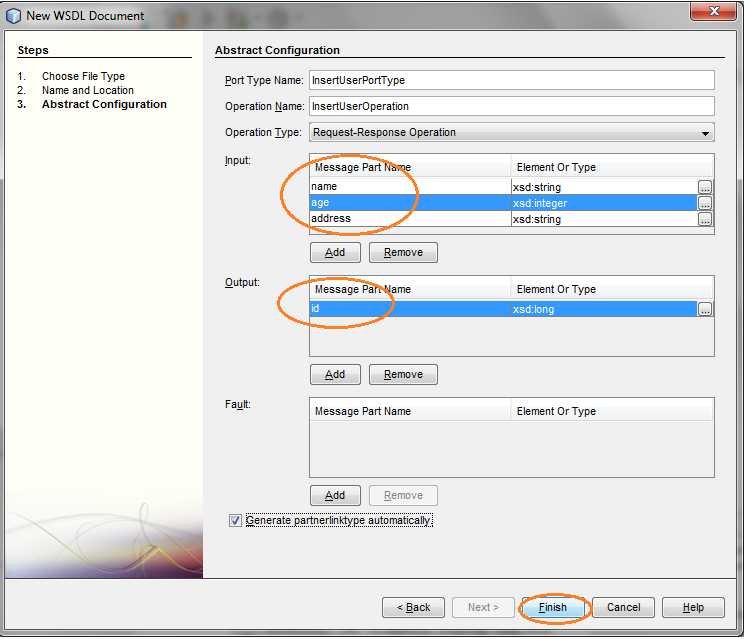
Trong bước này chúng ta điền JNDI name vào, chính là tên lúc cấu hình JDBC Resource trên server. Nhấn Finish.

Sau khi file DB\_insertUser.wsdl được tạo thì có một file .xsd với tên tương tự cũng được tạo. File này là một file XML schema, là file định nghĩa đối tượng trong XML.

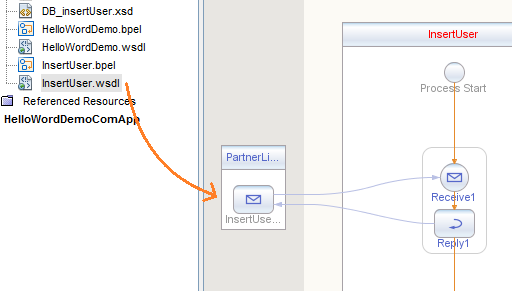


Tạo file .bpel với tên là **InsertUser.bpel** và một partnerlink wsdl **InsertUser.wsdl**

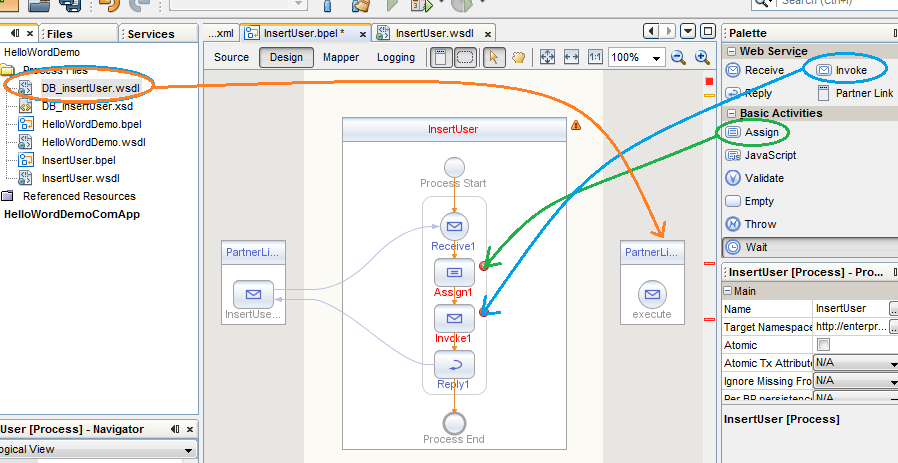
File InsertUser.wsdl được cấu hình với input là 3 biến: name, age và address. Biến trả về có tên là id. Hình dưới



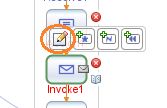
Kéo partnerlink insertUser.wsdl vào trong insertUser process ta được quy trình như hình dưới



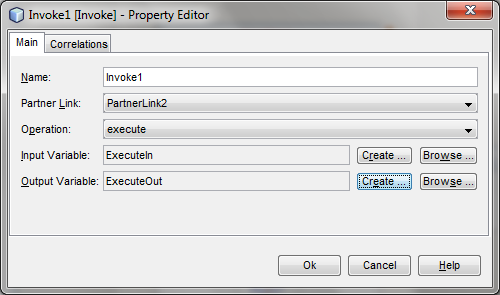
Kéo thả các thành phần như hình vẽ bên dưới.



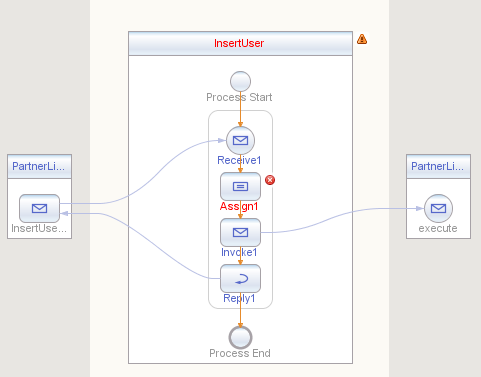
Click vào biểu tượng edit của Invoke. Invoke trong process có nghĩa là gọi dịch vụ.



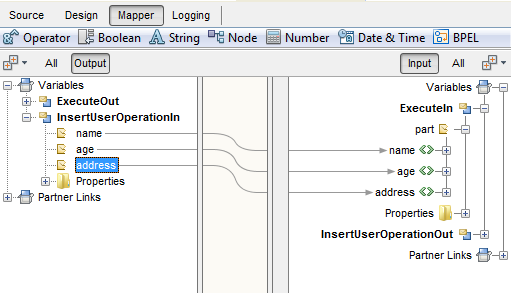
Cửa sổ edit hiện ra, cấu hình thông tin Partner Link, input , output.



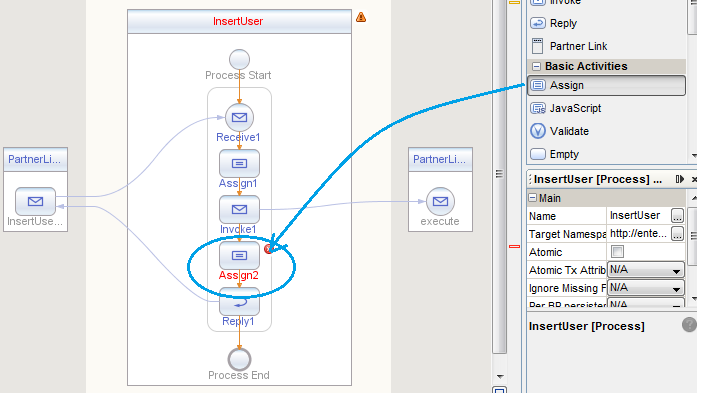
Cấu hình xong sẽ thấy mũi tên trỏ từ Invoke tới PartnerLink2.

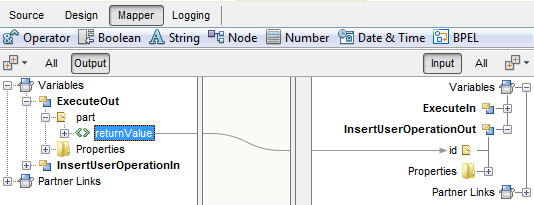


Double click vào Assign1 map dữ liệu như hình dưới. Thao tác này mục đích map dữ liệu vào cho function insert\_user.

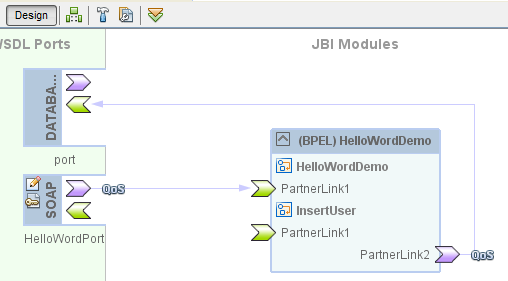


Tạo Assign2 để map dữ liệu trả về của quy trình.

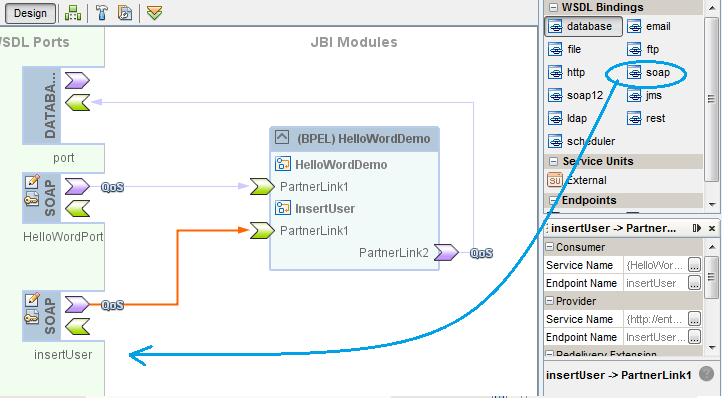




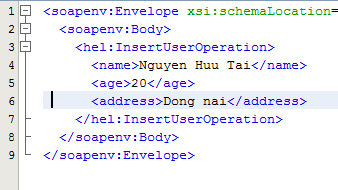
Build HellowordCommApp project sẽ thấy xuất hiện InsertUser process hiện ra như hình dưới.



Kéo thả WSDL Binding SOAP vào WSDL ports



Test quy trình với input như hình dưới.



Đây là kết quả trả về

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Body>

<m:InsertUserOperationResponse xmlns:m="HelloWordDemoComApp" xmlns:msgns="http://j2ee.netbeans.org/wsdl/HelloWordDemo/InsertUser">

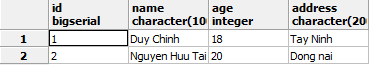
<id xmlns:msgns="http://j2ee.netbeans.org/wsdl/HelloWordDemo/DB\_insertUser"

xmlns:ns0="http://j2ee.netbeans.org/wsdl/HelloWordDemo/InsertUser" isNull="false">2</id>

</m:InsertUserOperationResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>



**Tổng kết**

Kết thúc demo này bạn biết được cách gọi một service trong BPEL module, cách lưu dữ liệu xuống database.

* + 1. **Kết hợp BPEL project và EJB project**

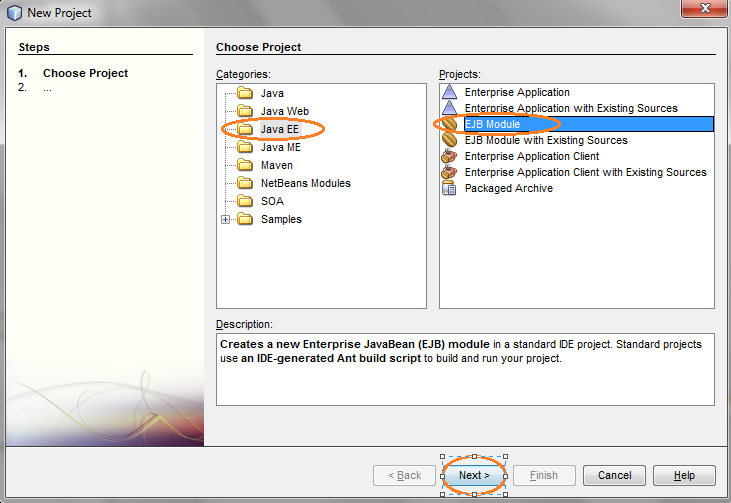
Như bạn cũng biết EJB là kỹ thuật tạo web service bằng ngôn ngữ java và điều tuyệt vời là ta có thể kết hợp BPEL với EJB.

Trong demo này tôi sẽ làm lại bài HelloWord nhưng sữ dụng BPEL module kết hợp với EJB module.

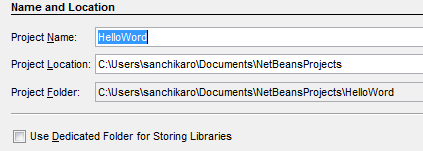
* Tạo EJB project

Vào New/New project.

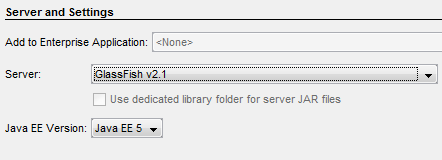
Cửa sổ New project hiện ra bạn chọn Java EE/EJB Module nhấn Next.



Bước tiếp theo đặt tên project là HelloWord nhấn Next.

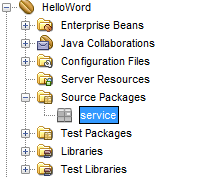


Trong mục Server and settings chọn server là GlassFish v2.1 nhấn Finish

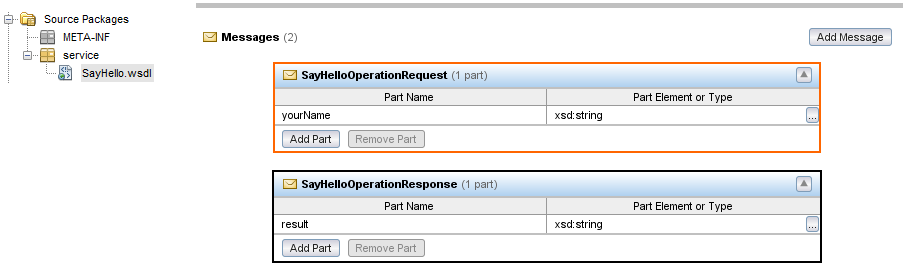


**Tạo webservice**

Tạo một package có tên là service.

****

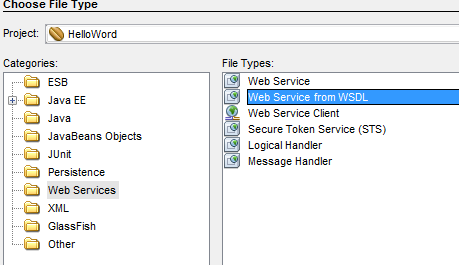
Tạo file WSDL có tên là SayHello. File này tương tự như partnerLink giống như trong BPEL module. Trong file này bạn cấu hình input message có tên là yourName kiểu là String, output message có tên là result kiểu là String.

\

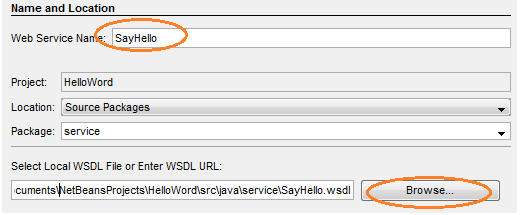
**Tạo web service từ file WSDL vừa tạo**

Click chuột phải lên project chọn New File.

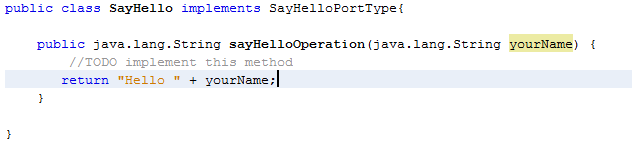
Cửa sổ New File hiện ra bạn chọn Web Services/Web Service from WSDL nhấn Next.



Tại thẻ input Web Service name bạn điền SayHello và nhấn Browse chỉ đường dẫn tới file WSDL vừa tạo nhấn Finish.

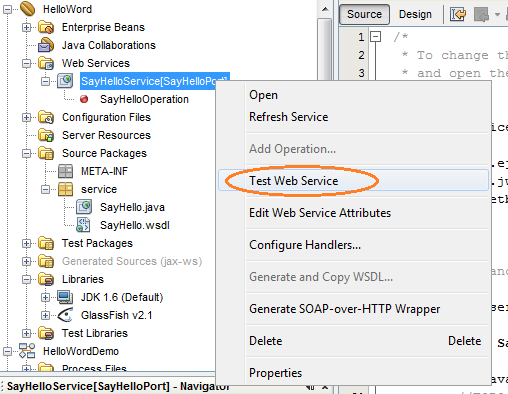


Sau khi webservice được tạo bạn edit phương thức SayHello như hình dưới.

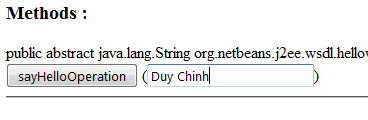


**Test webservice**

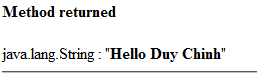
Sau khi build và deploy webservice, click chuột phải lên service muốn test chọn Test Web Service.



Giao diện test như hình dưới.

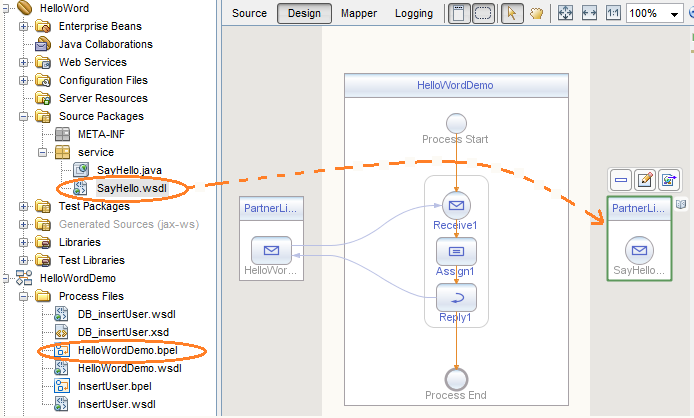


Kết quả test.



**Tạo BPEL project**

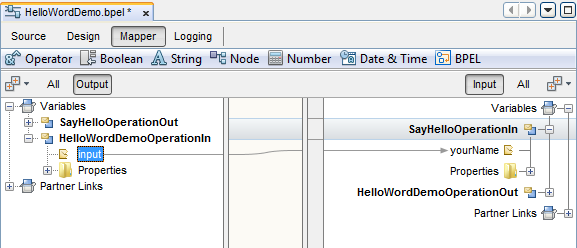
Mở BPEL process HelloWordDemo.bpel đã làm ở Demo đầu tiên. Kéo thả file WSDL sayHello.wsdl từ EIB project như hình bên dưới.

****

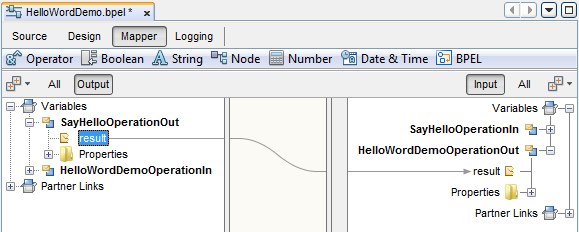
Kéo thẻ các thành phần như hình bên dưới.

****

Map lại Assign1 như sau



Map Assign2 như sau

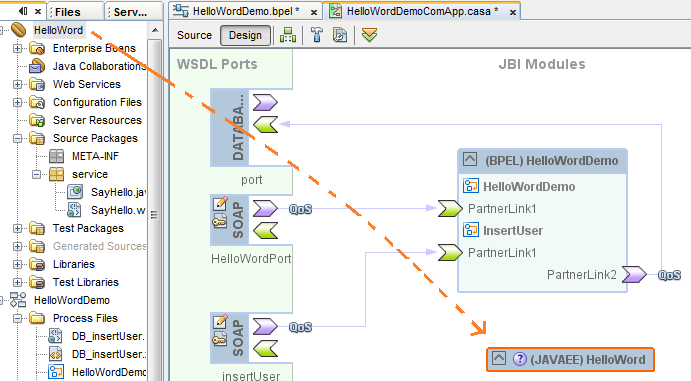


Build BPEL project.

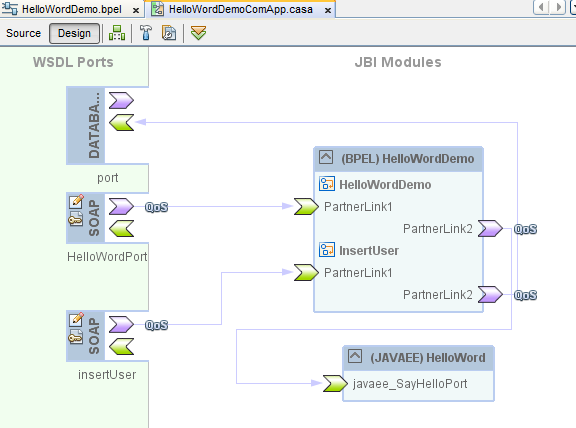
**Deploy và Test**

Chuyển qua HelloWordDemoComApp project kéo project EJB vào vùng trống của JBI module.

Sau đó build project.

****

Sau khi build



Bạn tiến hành test như hướng dẫn ở demo đầu tiên.

* Tổng kết

Composite Application có thể tích hợp nhìu JBI module lên đó. BPEL module là 1 loại JBI module và EJB module cũng là một JBI module. Giữa các module có thể trao đổi tương tác với nhau thông qua các partnerlink.

* 1. **Bài toán ứng dụng**
     1. **Phát biểu bài toán**

Xây dựng một hệ thống dùng để quản lý hoạt động của trường học bao gồm quản lý thông tin học sinh,quản lý thông tin giáo viên,quản lý việc thu tiền học phi.Đồng thời cung cấp chức năng gửi mail và tin nhắn SMS để trao đổi thông tin giữa nhà trường và phụ huynh.

* + 1. **Các chức năng cần xây dựng**

**Quản lý học sinh**

|  |  |
| --- | --- |
| **Mô tả** | **Yêu cầu chức năng** |
| Đây là chức năng quản lý học sinh trong nhà trường.Sử dụng chức năng này để theo dõi,quản lý các hoạt động liên quan đến học sinh. | Thêm mới học sinh. |
| Cập nhật thông tin học sinh. |
| Xem thông tin chi tiết học sinh. |
| Xóa học sinh. |
| Liệt kê danh sách và tìm kiếm học sinh theo những tiêu chí cho trước. |
| Cập nhật sổ điểm. |
| Xếp hạng học sinh. |

**Quản lý giáo viên**

|  |  |
| --- | --- |
| **Mô tả** | **Yêu cầu chức năng** |
| Đây là chức năng quản lý giáo viên của trường.Sử dụng chức năng này để theo dõi,quản lý các hoạt động liên quan tới giáo viên. | Thêm mới giáo viên. |
| Cập nhật thông tin giáo viên. |
| Xem thông tin chi tiết giáo viên. |
| Xóa giáo viên. |
| Tìm kiếm học sinh theo những tiêu chí cho trước. |
| Phân công giảng dạy theo các lớp. |

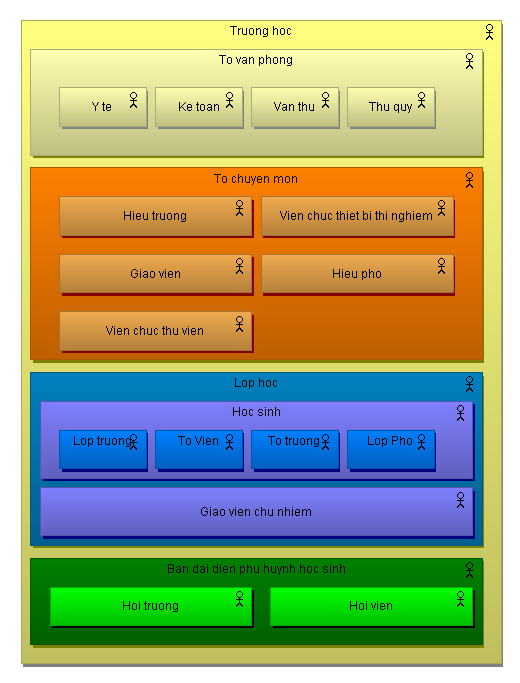
**Quản lý việc thu học phí**

|  |  |
| --- | --- |
| **Mô tả** | **Yêu cầu chức năng** |
| Đây là chức năng quản lý việc thu tiền học phí.Sử dụng chức năng này để theo dõi,quản lý các vấn đề liên quan tới học phí của học sinh. | Cập nhật tiền học phí của học sinh. |
| Xem chi tiết thông tin học phí của từng học sinh. |
| Liệt kê danh sách học sinh còn nợ học phí,đã hoàn thành học phí. |

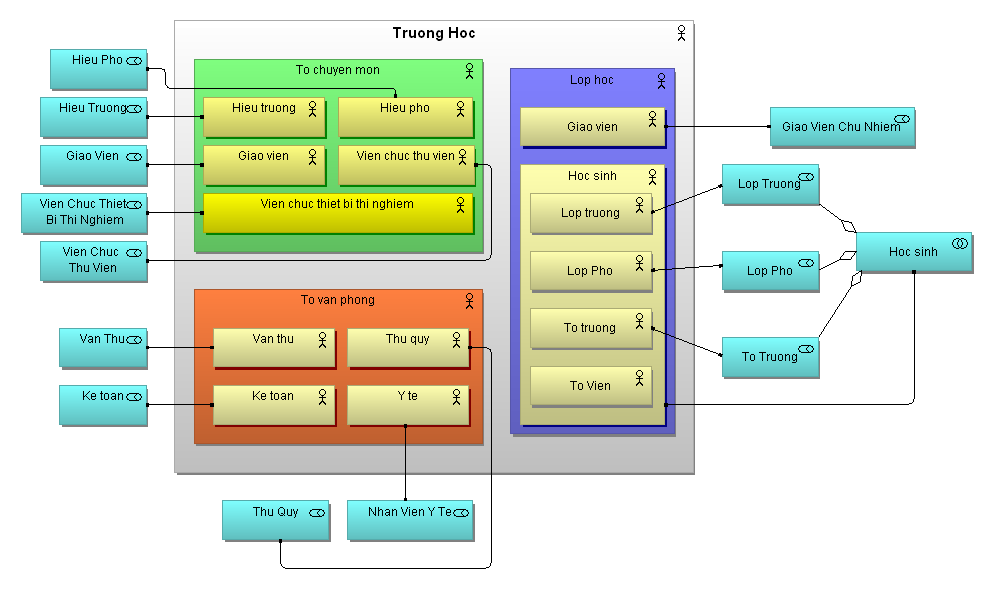
**Gửi mail và tin nhắn SMS**

|  |  |
| --- | --- |
| **Mô tả** | **Yêu cầu chức năng** |
| Đây là chức năng gửi mail và tin nhắn SMS.Sử dụng chức năng này để trao đổi thông tin giữa nhà trường và phụ huynh học sinh. | Cho phép gửi và nhận mail. |
| Cho phép gửi và nhận tin nhắn SMS. |

* + 1. **Mô hình kiến trúc thể hiện qua archimate**
* Bussiness layer



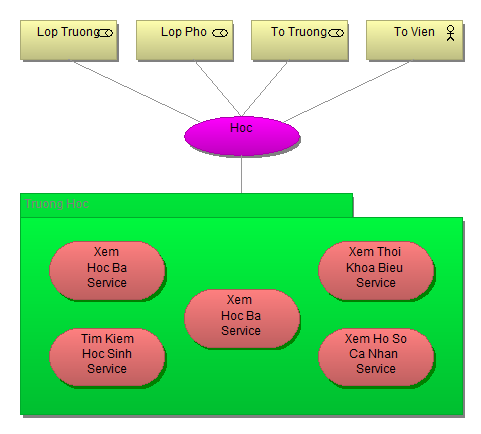
1. Bussiness Organisation structure view



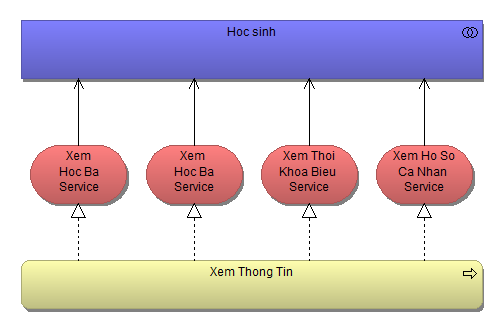
1. Bussiness Actor cooperation view



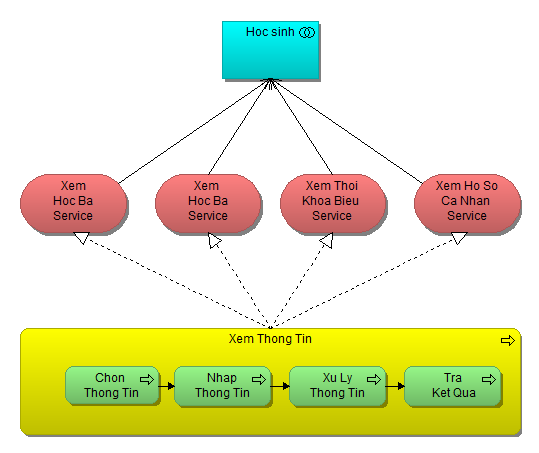
1. Business function view



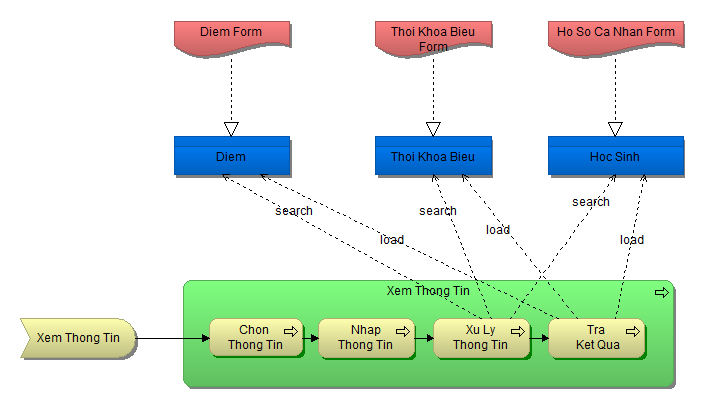
1. Business product view



1. Bussiness Service realisation view

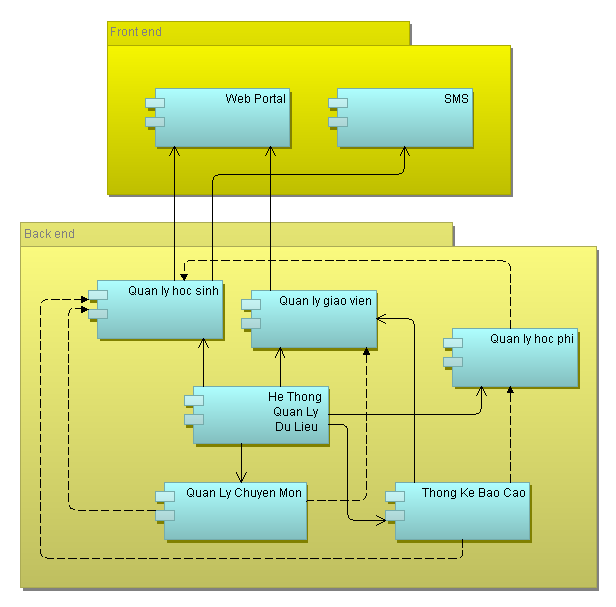


1. Business process cooperation view

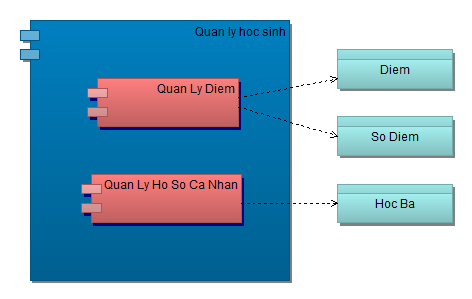


1. Bussiness process view

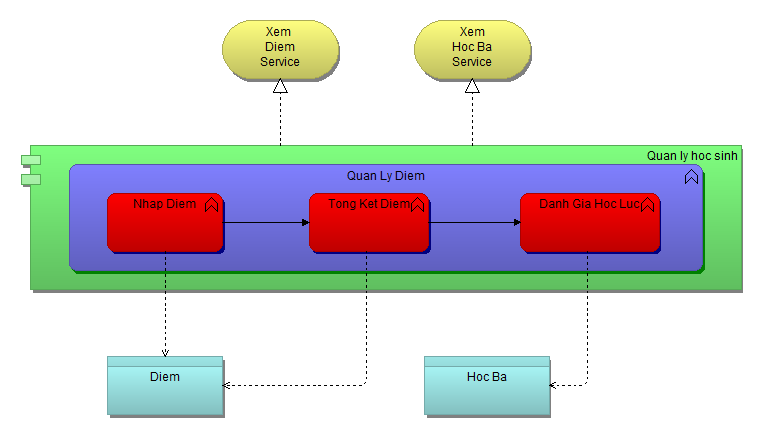
**Application Layer**



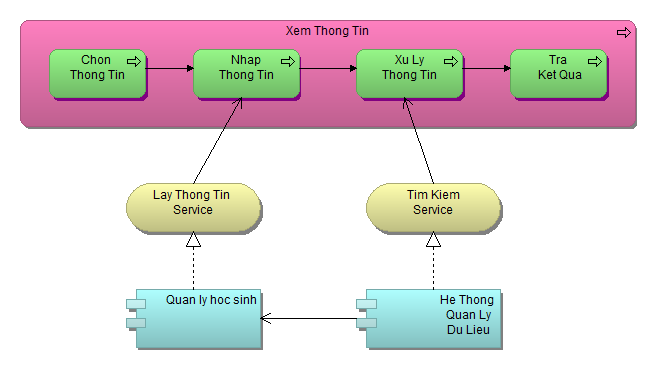
1. Application cooperation view



1. Application structure view

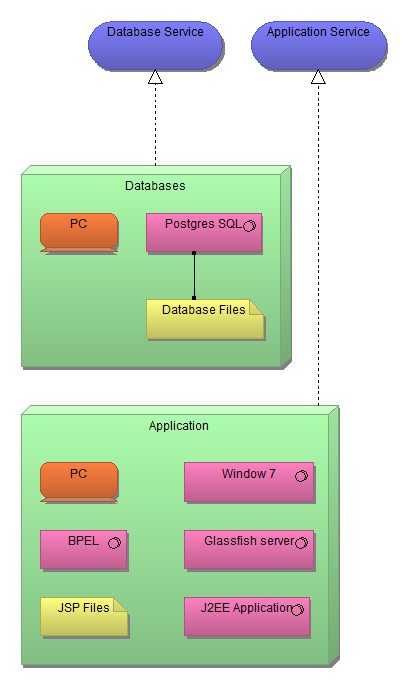


1. Application behavior view



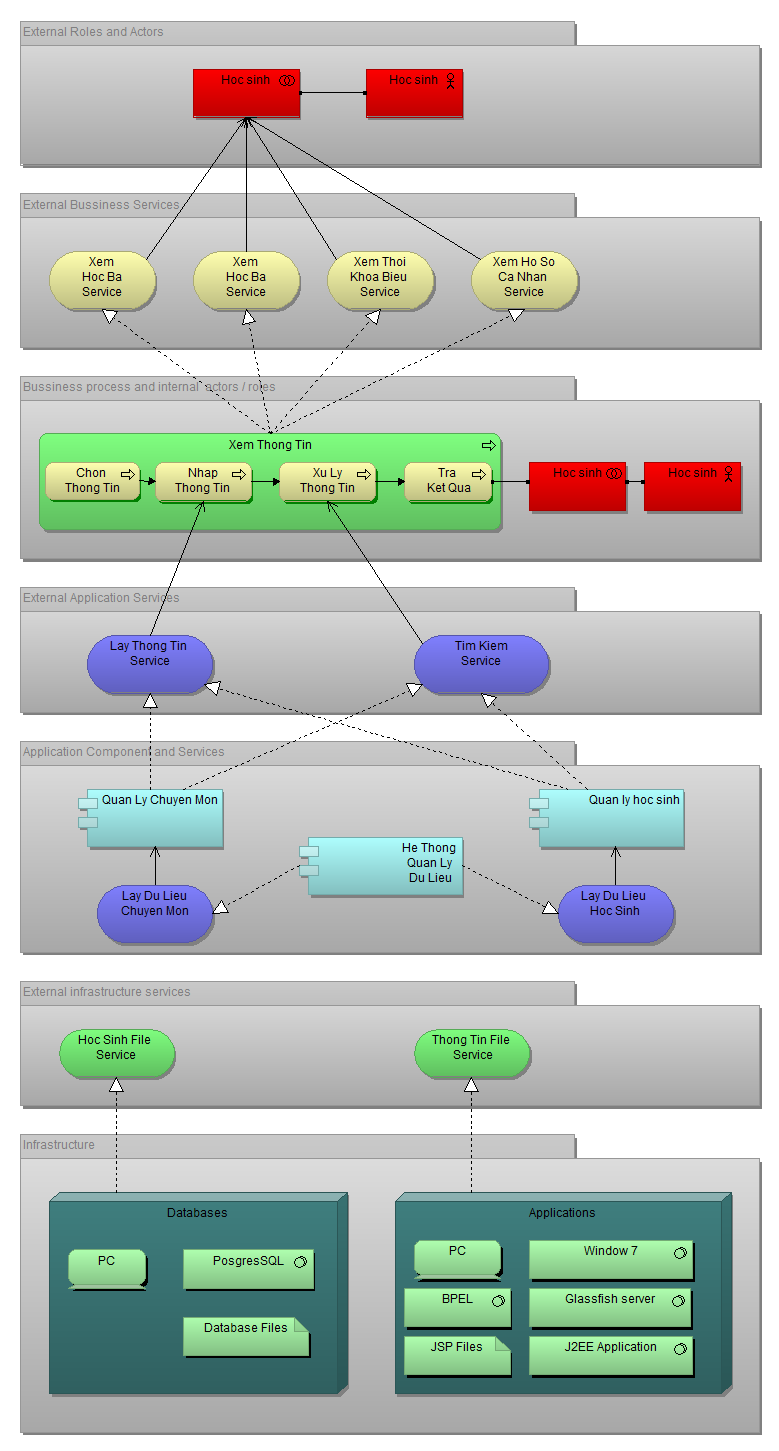
1. Application usage view

* Technology layer



1. Technical Infrastructure View

* Layer View



1. Layer view model

# KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN

* 1. **KẾT QUẢ ĐẠT ĐƯỢC**
* Biết được kiến thúc cơ bản về kiến trúc Enterprise.
* Nắm bắt phương pháp kiến trúc hệ thống theo ngôn ngữ Archimate.
* Hiện thực mô hình kiến trúc bài toán áp dụng.
  1. **HƯỚNG PHÁT TRIỂN**

Đề tài sẽ áp dụng các kiến thức đã tìm hiểu được áp dụng vào bài toán hệ thống quản lý trường trung học phổ thông.

# TÀI LIỆU THAM KHẢO

**[1]Marc Lankhorst**, Enterprise Architectureat Work*, lần thứ nhất, Springer,2005,3-540-24371-2.*

**[2] Hugo ter Doest,Maria-Eugenia Iacob,Marc Lankhorst,Diederik van Leeuwen,Robert Slagter**,Viewpoints Functionality and Examples*, bản cuối,Telematica Instituut,2004 .*

**[3]The Open Group**,ArchiMate® 1.0 Specification, *lần thứ nhất, The Open Group, 2009, 1-931624-80-1.*

**[4] Rob C. Thomas II** , Federal Enterprise Architecture , lần thứ nhất *, Federal Architecture Working Group (FAWG) , 2001 .*

**[5] Harmen van den Berg, Hans Bosma, Gertjan Dijk, Hans van Drunen, Jan**

**van Gijsen, Frank Langeveld, Joost Luijpers, Thé Nguyen, Ger Oosting,**

**Robert Slagter, Egon Willemsz** , ArchiMate Made Practical*, lần thứ hai , ArchiMate Foundation , 2007 .*