**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 : 32**

**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 Cong 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 Cong Song**

Giáo viên phản biện: **………………………**

Luận văn cử nhân được bảo vệ tại HỘI ĐỒNG CHẤM LUẬN VĂN CỬ NHÂN 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:**

Họ tên sinh viên: …………….. Phái: ………….

Ngày tháng năm sinh: ………………. Nơi sinh: …………….

Chuyên ngành: ……………………….

Địa chỉ: …………………………………………………………………..

Điện thoại liên lạc: ………………………………….

Email: ………………………………...

# LỜI CẢM ƠN

**🙞🏵🙜**

*Sau thời gian nghiên cứu luận văn, 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 Cong 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

**JDK J**ava **D**evelopment **K**it

Bộ nhân phát triển Java

**DAO D**ata **A**ccess **O**bject

Đối tượng truy xuất dữ liệu

**GUI G**raphics **U**ser **I**nterface

Giao diện người dùng

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

Ngôn ngữ mô hình hợp nhất

**API A**pplication **P**rogramming **I**nterface

Giao diện lập trình ứng dụng

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

Kiến trúc hướng dịch vụ

**EAI** **E**nterprise **A**pplication **I**ntegration

Ứng dụng tích hợp thương mại

**XML** E**X**tensible **M**arkup **L**anguage

Ngôn ngữ đánh dấu mở rộng

**SCA** **S**ervice **C**omponent **A**rchitecture

Kiến trúc thành phần dịch vụ

**WSDL** **W**eb **S**ervices **D**escription **L**anguage

Ngôn ngữ miêu tả cho web service

**SOAP** **S**imple **O**bject **A**ccess **P**rotocol

Giao thức truy xuất đối tượng đơn giản

**HTTP** **H**yper**T**ext **T**ransfer **P**rotocol

Giao thức truyền tải siêu văn bản

# DANH SÁCH CÁC THUẬT NGỮ TIẾNG ANH

**Business process** quy trình nghiệp vụ

**Message** Thông điệp

**Message metadata** Dữ liệu thông điệp

**Plug** Gắn vào

**Adapter** Bộ tương thích

**Workflow** Luồng công việc

**Data caching** Lưu trữ dữ liệu

**Data collection** Tập hợp dữ liệu

# MỤC LỤC

🙞🏵🙜

[BỘ GIÁO DỤC VÀ ĐÀO TẠO i](#_Toc263061233)

[TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM i](#_Toc263061234)

[KHOA CÔNG NGHỆ THÔNG TIN i](#_Toc263061235)

[TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM ii](#_Toc263061236)

[KHOA CÔNG NGHỆ THÔNG TIN ii](#_Toc263061237)

[LỜI CẢM ƠN i](#_Toc263061238)

[DANH SÁCH CHỮ VIẾT TẮT ii](#_Toc263061239)

[DANH SÁCH CÁC THUẬT NGỮ TIẾNG ANH iii](#_Toc263061240)

[MỤC LỤC iv](#_Toc263061241)

[DANH MỤC CÁC HÌNH vi](#_Toc263061242)

[TÓM TẮT vii](#_Toc263061243)

[Tên đề tài vii](#_Toc263061244)

[Nội dung nghiên cứu vii](#_Toc263061245)

[Hướng tiếp cận và giải quyết vấn đề vii](#_Toc263061246)

[Một số kết quả đạt được viii](#_Toc263061247)

[CHƯƠNG 1. MỞ ĐẦU 1](#_Toc263061248)

[**1.1. LÝ DO CHỌN ĐỀ TÀI** 1](#_Toc263061249)

[**1.2. MỤC TIÊU ĐỀ TÀI** 1](#_Toc263061250)

[**1.3. PHẠM VI NGHIÊN CỨU** 1](#_Toc263061251)

[CHƯƠNG 2. TỔNG QUAN 3](#_Toc263061252)

[**2.1. ĐẶT VẤN ĐỀ** 3](#_Toc263061253)

[CHƯƠNG 3. NỘI DUNG NGHIÊN CỨU 4](#_Toc263061254)

[**3.1. Giới thiệu cơ bản về Enterprise Architecture** 4](#_Toc263061255)

[**3.1.1. Các thuật ngữ chuyên môn** 4](#_Toc263061256)

[**3.1.2. Tại sao phải có kiến trúc enterprise** 5](#_Toc263061257)

[**3.1.3. Quy trình kiến trúc hệ thống** 6](#_Toc263061258)

[**3.1.4. Communication of Enterprise Architectures** 6](#_Toc263061259)

[**3.1.5. Các phương pháp kiến trúc** 12](#_Toc263061260)

[**3.1.6. Các framework hổ trợ** 13](#_Toc263061261)

[**3.1.7. Các ngôn ngữ kiến trúc hệ thống** 15](#_Toc263061262)

[**3.2. Archimate language** 25](#_Toc263061263)

[**3.2.1. ArchiMate là gì** 26](#_Toc263061264)

[**3.2.2. Tại sao dùng ArchiMate** 26](#_Toc263061265)

[**3.2.3. Những lợi ích chính của ArchiMate** 29](#_Toc263061266)

[**3.2.4. Các khái niệm chính trong ngôn ngữ** 29](#_Toc263061267)

[**3.2.5. Những ký hiệu của ngôn ngữ** 31](#_Toc263061268)

[**3.2.6. Kiến trúc tổng quát ngôn ngữ Archimate** 33](#_Toc263061269)

[**3.2.7. Tầng Business** 35](#_Toc263061270)

[**3.2.8. Tầng Application** 55](#_Toc263061271)

[**3.2.9. Tầng Technology** 65](#_Toc263061272)

[**3.2.10. Viewpoint** 69](#_Toc263061273)

[**3.2.11. So sánh Archimate và UML** 116](#_Toc263061274)

[**3.3. SOA** 116](#_Toc263061275)

[**3.3.1. Giới thiệu** 116](#_Toc263061276)

[**3.3.2. Chi tiết** 116](#_Toc263061277)

[**3.4. Bài toán ứng dụng** 116](#_Toc263061278)

[**3.4.1. Phát biểu bài toán** 116](#_Toc263061279)

[**3.4.2. Các chức năng cần xây dựng** 116](#_Toc263061280)

[**3.4.3. Mô hình kiến trúc thể hiện qua archimate** 116](#_Toc263061281)

[**3.4.4. Mô hình UML thể hiện chi tiết thiết kế** 116](#_Toc263061282)

[**3.4.5. Thiết kế cơ sở dữ liệu** 116](#_Toc263061283)

[**3.4.6. Chức năng quản lý giáo viên** 117](#_Toc263061284)

[**3.4.7. Chức năng quản lý học sinh** 117](#_Toc263061285)

[**3.4.8. Chức năng quản lý tiền thu học phí** 117](#_Toc263061286)

[CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN 118](#_Toc263061287)

[**4.1. KẾT QUẢ ĐẠT ĐƯỢC** 118](#_Toc263061288)

[**4.2. HƯỚNG PHÁT TRIỂN** 118](#_Toc263061289)

[TÀI LIỆU THAM KHẢO 119](#_Toc263061290)

[PHỤ LỤC 120](#_Toc263061291)

# DANH MỤC CÁC HÌNH

**Hình** **Trang**

[Hình 3-1 Sự truyền thông giữa Enterprise Architects với các Stakeholders 4](#_Toc262718291)

[Hình 3-2 Vòng đời của quy trình kiến trúc hệ thống 6](#_Toc262718292)

[Hình 3-3 TOGAF architecture development cycle 7](#_Toc262718293)

[Hình 3-4 The Zachman Framework (Zachman 1987) 8](#_Toc262718294)

[Hình 3-5 TOGAF (based on The Open Group 2002). 8](#_Toc262718295)

[Hình 3-6 MDA framework. 9](#_Toc262718296)

[Hình 3-7 IDEF0 representation. 10](#_Toc262718297)

[Hình 3-8 Example model in BPMN 10](#_Toc262718298)

[Hình 3-9 Example of a business process model in Testbed. 11](#_Toc262718299)

[Hình 3-10 Events, functions and control flows in ARIS. 11](#_Toc262718300)

[Hình 3-11 Ví dụ một mô hình UML 12](#_Toc262718301)

# TÓM TẮT

## Tên đề tài

**“Tìm hiểu về Archimate và áp dụng triển khai hệ thống Quản lý trường trung học phổ thông”**

## Nội dung nghiên cứu

Enterprise Aplication

Enterprise Architecture

Archimate

SOA

Web serivce

BUS

XML

Truong Hoc

## Hướng tiếp cận và giải quyết vấn đề

Nghiên cứu các tài liệu kỹ thuật liên quan.

Tiếp cận thực tế quy trình quản lý một trường trung học phổ thông.

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

## Một số kết quả đạt được

………………………………………….

# MỞ ĐẦU

* 1. **LÝ DO CHỌN ĐỀ TÀI**

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.

Portal đã không phải là một khái niệm xa lạ. Tuy nhiên để xây dựng được một ứng dụng portal theo đúng nghĩa không phải là điều dễ dàng.

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

* 1. **MỤC TIÊU ĐỀ TÀI**

Xây một dựng ứng dụng portal có có khả năng tùy biến cao, áp dụng được cho tất cả các trường trung học phổ thông .

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 Archimate language ứng dụng vẻ mô hình hệ thống.
3. 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**

Hiện tại, ở các trường trung học phổ thông công việc quản lý của nhà trường vẫn còn làm thủ công hoặc nếu có áp dụng công nghệ thông tin thì cũng chỉ áp dụng ở một vài khâu với nhiều phần mềm riêng lẻ khác nhau. Rất khó khăn trong việc tổng kết , thống kê số liệu một cách chính xác và nhanh chống.

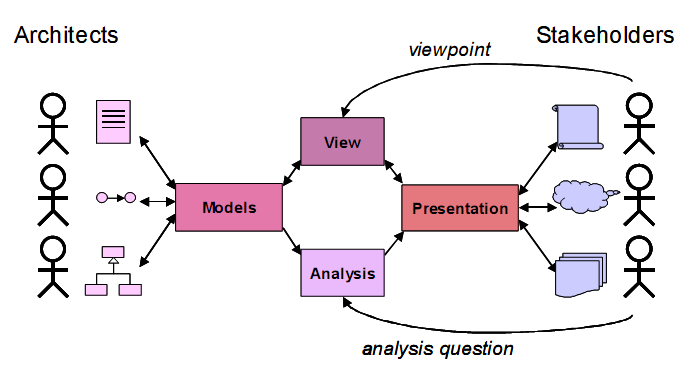
# NỘI DUNG NGHIÊN CỨU

* 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 rỏ về hệ thống theo cách nhìn phù hợp với họ , cái đó người ta gọi là các viewpoint.



1. Sự truyền thông giữa Enterprise Architects với các Stakeholders

**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 chặc chẻ, các phương thức và những mô hình mà được sử dụng trong thiết kế và sự hiện thực kiến trúc, quy trình nghiệp vụ, thông tin hệ thống , và cơ sở hà 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 cở 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 sảy 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 sảy 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 mốc , 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 sảy ra bên ngoài doanh nghiệp và nằm ngoài tằm kiểm soát của daonh 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.

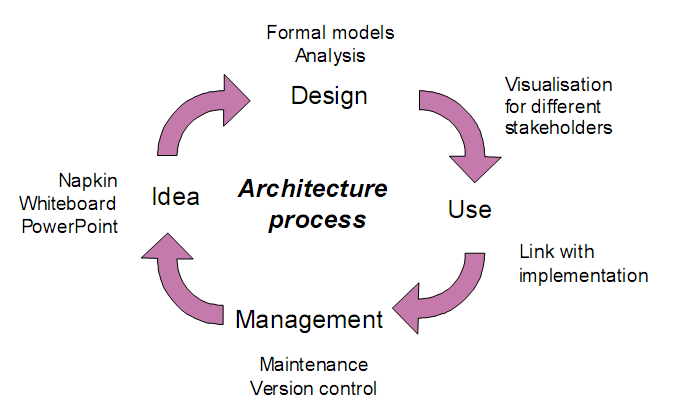
Ngoài việc cung cấp cái nhìn tổng thể thì kiến trúc enterprise còn được sử dụng để ước lượng , đánh giá quá trình chuyển đổi từ hiện tại đến tương lai. Nó cung cấp một phương pháp để đánh giá tác động của những thay đổi kiến trúc đến chất lượng sản phẩm và cả khía cạnh số lượng, ví dụ như hiệu năng sản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

* + 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 stackeholder . Tùy vào từng stackeholder khác nhau mà có tài liệu phù hợp với stackehoder 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 vẻ , các tài liệu thiết kế lại thành một tập version và quản lý từng version thiết kế. Và cứ như vậy version của các bảng thiết kế cứ phát triển dần dần.



1. Vòng đời của quy trình kiến trúc hệ thống
   * 1. **Communication of Enterprise Architectures**

Vai trò của sự truyền thông:

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 enterprise architectect.

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 trong việc truyền thông. Một ví dụ về architecture description, 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 đối 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 người yêu cầu và người phát triển, như là một phần của bản hợp đồng…Architecture description 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.

Trong những phần tiếp theo chúng ta sẽ thảo luận chi tiết các vấn đề liên quan đến sự truyền thông, những mục đích cần đạt được từ việc truyền thông và những nguyên tắc hỗ trợ để đạt được kết quả mong muốn từ sự truyền thông hiệu quả.

Sự phát triển hệ thống được xem như là quá trình trao đổi thông tin:

Về bản chất, chúng ta đều thấy rằng sự phát triển hệ thống cũng như là quá trình trao đổi *knowledge*, do đó những *conversation( cuộc hội thảo)* được sử dụng để chia sẽ và tạo nên những *knowledge* vừa gắn liền với hệ thống đang được phát triển, vừa gắn liền sự phát triển chính bản thân nó. Tiếp theo đây chúng ta sẽ làm rõ hơn về vấn đề này.

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 domain experts, sponsors, architects, engineers, business analysts…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 knowledge 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 *concert* của stackholder.

+***Concert***: Là mối quan tâm của stackholder về những vấn đề liên quan đến architecture description của hệ thống, kết quả từ những mục đích của stackholder, và vai trò hiện tại và tương lai của hệ thống trong mối quan hệ với những mục đích này.

System Development Knowledge

System development community thường mang knowledge về hệ thống đang được phát triển, sự thông tin xuất hiện bên trong System development community thực chất là để nhắm đến việc tạo, phổ biến knowledge này. Dựa trên concern của họ, những stackholder này sẽ được hướng sự quan tâm đến những topic(chủ đề) khác nhau gắn liền với hệ thống đang được phát triển.

Trong suốt sự phát triển của hệ thống, những thành viên của System development community sẽ tạo và trao đổi knowledge gắn liền với những chủ đề khác nhau. Đầu tiên chúng ta có phân biệt giữa target domain-gắn liền với hệ thống đã và đang được phát triển và project domain gắn liền với chính bản thân quy trình phát triển hệ thống:

Đối với knowledge của của hai domain này, bằng cách quan tâm đến những topic có thể có chúng ta có có thể làm sáng tỏ những knowledge này. Những topic này có thể được nhận dạng bằng những đặc trưng sau:

Perspective: Điển hình như một hệ thống, có thể được xem xét ở nhiều khía cạnh ví dụ như:

Khía cạnh business, application, infrastructure

Khía cạnh luận lý, khía cạnh vật lý của một hệ thống

Scope: Từ một domain cho trước ,như là một hệ thống, chúng ta có thể nhận diện một vài giới hạn khi tiếp cận domain như: enterprise-wide, department-specific, task-specific…

Design chain: Khi xem xét thiết kế của một vài artifact, chúng ta có thể phân biệt giữa:

Purpose: đâu là mục đích artefact cần.

Functinality: artefact sẽ cung cấp những chức năng gì cho môi trường.

Design: Nó hiện thực chức năng này như thế nào.

Quality: nó thực thiện những chức năng này có tốt không

Costs: Chi phí để đưa ra một artefact

Historical Perspective: Với một artifact được thiết kế sẵn, một người nào đó có thể nghĩ đến những phiên bản khác nhau artefact này đã được thiết kế trước đó.

Abstraction level: khi quan tâm đến một domain, một người nào đó có thể quan tâm đến một vài cấp độ trừu tượng hoá. Những sự trừ tượng hoá đa dạng có thể được tách biệt ở đây, ví dụ như: type-instance, generalisation/is-a, encapsulation, và ẩn sự thực thi chi tiết bên dưới.

Explicitness of Knowledge

Explicit knowledge là những knowledge có thể biểu diễn được bằng ngôn ngữ hay bằng mô hình, nó phân biệt với tacit knowledge, những knowledge khó có thể được biểu diễn một cách rõ ràng được. Trong ngữ cảnh truyền thông của chúng ta, chúng ta sẽ tập trung vào những knowledge có thể mô tả rõ ràng được hay còn được gọi là explicit knowledge.

System description là một dạng thể hiện cơ bản của explicit knowledge. Chúng ta nhận diện những dimension của explicit knowledge cho sự biểu diễn của những system development knowledge như sau:

***Formality:*** Sự thoả thuận về loại ngôn ngữ được sử dụng để mô tả knowledge, có thể là một ngôn ngữ chính thức như là những ngôn ngữ được định nghĩa rõ ràng chính xác, hay ngôn ngữ không chính thức như ngôn ngữ tự nhiên, hình ảnh minh hoạ…

***Quantifiability:*** Những khía cạnh khác nhau của một artefact, có thể là một phần của target hoặc project domain,phải được xác định về volume, capacity, workload, effort, resource usage, time duration, frequency…

***Executability:*** sự biểu diễn knowledge rõ ràng đủ để có thể thấy được khả năng thực thi của dự án.

***Comprehensebility:*** Khả năng lĩnh hội của người nghe

***Completeness:*** knowledge presentation có thể hoàn chỉnh, hay chưa hoàn chỉnh, điều này liên quan đến knowledge topic.

Transformations of Knowledge

Trong suốt sự phát triển của hệ thống, knowledge về hệ thống sẽ phát triển không ngừng. Sau khi đã đạt được sự thấu hiểu nhất định về hệ thống, những bản thiết kế được vẽ ra, những view được chia sẽ, những ý kiến được nhào nặn lại, quyết định thiết kế được đưa ra…Tất cả những điều này dẫn đến sự biến đổi( transformation) của “knowledge state”. Sự biến đổi này thường được mang về từ những conversation.

Cả hai presentation và actor đều có thể được xem là những đối tượng mang những knowledge topic. Như vậy presentation và actor đều là *knowledge carriers*

Những knowledge được biểu diễn trong một development community có phát triển theo một số trạng thái. Đầu tiên, knowledge cần được giới thiệu vào community điều này có thể là tạo những knowledge từ bên trong, hay import những knowledge từ bên ngoài vào community. Sau khi knowledge đã được giới thiệu, nó có thể được chia sẽ giữa các thành viên trong community.Việc chia sẽ knowledge giữa những thành viên có thể trải qua ba giai đoạn chính :

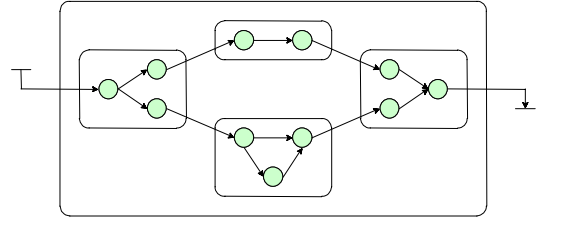
***Aware:*** Một actor có thể trở thành một aware của knowledge khi được chia sẽ bởi một actor khác.

***Agreed:*** Khi knowledge được chia sẽ, một actor có thể thể hiện suy nghĩ của mình về knowledge đó, và quyết định có đồng ý hay không với knowledge được chia sẽ này không.

***Committed:***Những actor đã đồng ý với knowledge này có thể quyết định để thực sự giao phó cho knowledge này hay không.

Conversation Strategies

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



Knowledge Goals

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

Xác định Conversation strategy

Mỗi conversation điều được diễn ra để đạt được một knowledge goal nào đó, có nghĩa là một trang thái nào đó của knowledge mà một conversation nhắm đến, và để đạt được những mục đích này, conversation sẽ phải theo một conversation strategy nào đó.

Một conversation thường phải đặt trong một ngữ cảnh nào đó, và điều này có ảnh hưởng nhất định đến conversation. 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 conversation

Complexity: Liên quan đến tài nguyên cần thiết, knowledge cần được trao đổi…Yếu tố này có thể tác động đến conversation strategy. 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 knowledge cần thảo luận, sự phức tạp của công nghệ cần sử dụng…

Uncertainty: Để xác định được một conversation strategy, trước tiên chúng ta cần phải xác định được hoàn cảnh hiện tại, đưa ra những nhận định về knowledge goal, trạng thái của knowledge, xác định tài nguyên sẵn có…Trong quá trình thực thi conversation, những sự xác định này có thể sai, hoặc không chắc chắn.

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

Execution plan: Một conversation có thể gồm nhiều sub-conversation, Mỗi sub-conversation này thường tập trung vào một sub-goal, nhưng tất cả chúng đều hướng về mục đích chung của conversation. Execute plan của một conversation bao gồm một tập hợp những sub-conversation với sự thực thi theo một kế hoạch định trước

Description languages: Ngôn ngữ được sử dụng trong conversation

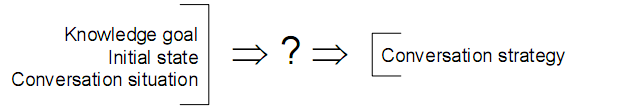
Media: Loại phương tiện được sử dụng trong suốt một conversation

Cognitive mode: Yêu cầu cách mà những knowledge được tập hợp hoặc xử lý bởi yêu cầu cua 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 conversation strategy phù 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 lập đi lập 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.

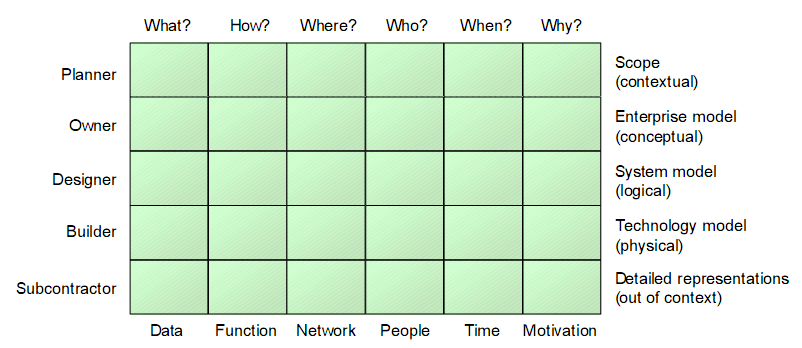
**UMM** - UN/CEFACT Modelling Methodology . Nó là một quy trình hoạt động gia tăng và là phương pháp xây dựng mô hình thông tinh. Business Collaboration Framework (BCF) là một framework đang được phát triển và sẽ là chuyên ngành của UMM nhằm xác định sự trao đổi thông tin bên ngoài của tổ chức và cá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



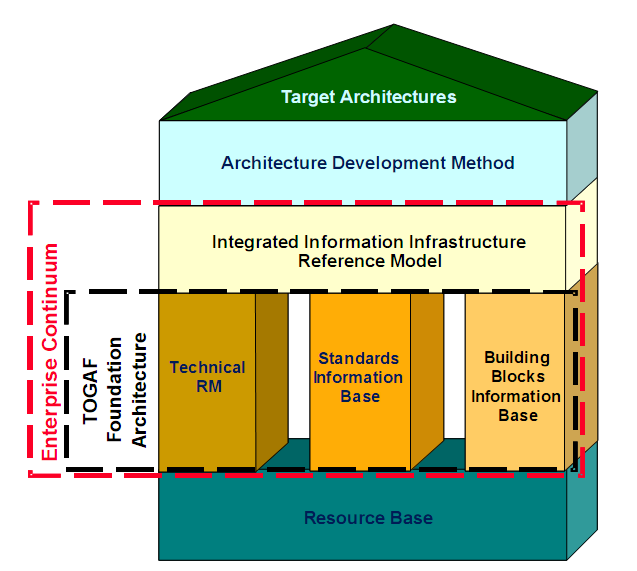
1. TOGAF architecture development cycle
   * 1. **Các framework hổ trợ**

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



1. The 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 của 1 ngôn ngữ chung đc SD để thực hiện mô hình mẫu khó và phân tích chúng. (xem http://www.idef.com/ and Mayer et al. 1995, IDEF 1993, Menzel and Mayer 1998). IDEF ( *sự xác định* *việc sx bằng máy điện toán đc hợp nhất lại* (ICAM)) nhóm phương pháp có nguồn gốc vũ trang. Ban đầu, chúng đc phát triển bởi lực lượng ko quân US về chương trình *Sự xác định* *SX bằng máy điện toán đc hợp nhất lại* (ICAM). Số lượng người tham gia trong cuộc họp về nhóm người SD IDEF là bằng chứng của việc người SD rộng rãi IDEF.

Hiện tại, có 16 phương pháp IDEF. Trong những pp này, IDEF0, IDEF3 và IDEF1X ( “trung tâm”) thì đc SD phổ biến nhất. Tầm xa của chúng bao trùm:

Mô hình chức năng, IDEF0: ý tưởng trước IDEF0 là mô hình hóa những yếu tố đang kiểm soát việc thực hiện 1 chức năng, người trình chiếu chức năng này, những object hay dữ liệu đc thu nạp và đc sx bởi chức năng này và những mối quan hệ giữa những chức năng của quá trình diễn biến ( share nguồn và những phần phụ thuôc).

Mô hình tiến trình, IDEF3: IDEF3 giữ lại tiến trình cv của 1 quá trình diễn biến thông qua tiến trình theo sau biểu đồ. Những tiến trình này biểu thị 1 cách liên tục cho đến khi tiến trình đc trình chiếu bởi 1 tổ chức nào đó, những quyết định hợp lý, những kịch bản khác mô tả cho việc trình chiếu cùng những chức năng của 1 qt diễn biến nào đó, và có thể phân tích và cải thiện qt công việc.

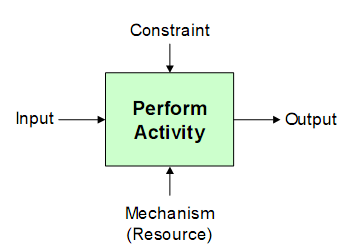
Dữ liệu chức năng, IDEF1X: IDEF1X đc SD để tạo ra những mô hình dữ liệu hợp logic và những mô hình dữ liệu vật chất thông qua ý nghĩa của những biểu đồ chức năng hợp logic, nhiều biểu đồ vùng subiect IDEF1X hợp logic và nhiều biểu đồ vật chất.

Có 5 yếu tố cho mô hình chức năng IDEF0: *activity* (hoặc tiến trình) được biểu diễn bởi những cái hộp, *inputs, outputs, constrains( sự hạn chế)* hoặc sự kiểm soát về những hoạt động, và *mechanisms( cơ cấu, kỹ thuật)* chúng thực hiện những hoạt động. *Inputs*, kiểm soát, những mũi tên *Output* và *mechanism* cũng có liên quan đến ICOMs. Mỗi hoạt động và ICOMs có thể đc phân tích (hoặc phá vỡ) thành nhiều mức chi tiết để để phân tích. Cơ chế phân tích cũng đc chỉ ra thành 1 kỹ thuật chức năng cho những đơn vị của cách vận hành IDEF3.

Phương pháp thu hút việc mô tả tiến trình IDEF3 cung cấp 1 cơ chế cho việc thu thập và cung cấp tiến trình. Có 2 mô hình mô tả IDEF3: tiến trình theo sau biểu đồ và trạng thái chuyển tiếp object biểu đồ hệ thống network. 1 tiến tình theo sau việc mô tả kiến thức thu thập đc của “ cách chúng làm việc” trong 1 tổ chức…, việc mô tả điều gì sẽ xảy ra đối với 1 phần nào đó bởi vì nó theo sau qua qua sự liên tục của những tiến trình SX. Trạng thái chuyển tiếp thông qua hệ thống network của những object này tổng kết lại những sự chuyển tiếp của 1 oject có thể chấp nhận có thể bị thay đổi lón thông qua 1 tiến trình đặc biệt.

Hệ thống IDEF cung cấp sự đồng tình cho việc mô hình hóa của vài 1 vài quan điểm thuộc về kiến trúc. Tuy nhiên, ko có những kỹ thuật giao tiếp giữa những mô hình này. Sự thật rằng chúng bị tách ra gây cản trở cho những hình dung của tất cả những mô hình như những yếu tố có liên quan với nhau của 1 hệ thống thuộc về kiến trúc. Điều này cũng có nghĩa rằng 1 sự thay đổi giữa những quan điểm là ko thể.

IDEF được SD rộng rãi trong CN. Điều này chỉ ra rằng nhu cầu của những người SD trong vòng giới hạn có thể chấp nhận đc, nó thì an toàn. Hệ thống IDEF là chủ đề cho 1 tiến trình liên tục của việc phát triển và cải thiện. Còn nữa, IDEF0, IDEF1X,IDEF3 là ngôn ngữ ổn định và cứng nhắc hơn, và IDEF0, IDEF1X đã đc xuất bản như 1 tiêu chuẩn của Viện tiêu chuẩn và kỹ thuật quốc gia.



1. IDEF0 representation.

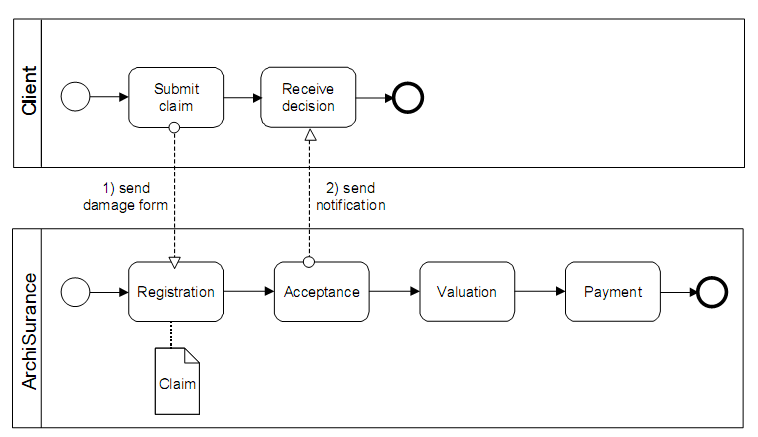
**Business Process Modelling Notation (BPMN)** –

Chú giải mô hình hóa tiến trình công việc (BPMN) là 1 trong những tiêu chuẩn đang đc phát triển bởi tổ chức khởi xướng việc quản lý tiến trình cv (BPMI). BPMI là 1 tổ chức phi lợi nhuận, mà mục đích của họ là: (1) chỉ rõ những tiêu chuẩn mở cho việc thiết kế tiến trình, (2) ủng hộ những nhà cung cấp và SD của những kỹ thuật và công cụ của việc quản lý tiến trình cv. Nhiều tổ chức liên quan trong việc mô hình hóa và quản lý tiến trình cv đc liên quan trong những hoạt động của BPMI. Việc phát triển khác bởi BPMI bao gồm Phát ngôn mô hình hóa tiến trình cv (BPML), 1 XML đc dựa trên ngôn ngữ meta cho việc trao đổi những mô hình tiến trình cv, và BPQL, 1 ngôn ngữ truy vấn cho tiến trình cv.

Tiêu chuẩn BPMN chuyên môn hóa 1 ký hiệu đồ họa thì đáp ứng như 1 điều cơ bản thông thường cho 1 mô hình tiến trình cv và lời phát ngôn có tính chất thực thi khác nhau. Việc làm bản đồ từ BPMN đến những cái khác, BPML và BPEL4WS ( Ngôn ngữ có tính chất thực thi tiến trình cv cho những dịch vụ Web) đã bị từ chối. Phiên bản 1.0 của BPMN ra mắt vào tháng 8/2003. Chẳng hạn như những chú giải của tiến trình cv đã đc xem: biểu đồ hoạt động UML, tiến trình cv UML EDOC, IDEF, ebXML BPSS, biểu đồ ADF, RosettaNet, LOVeM, và chủ tọa tiến trình sự kiện (EPCs).

Vì gần đây bị ám chỉ, BPMN bị hạn chế đến mô hình hóa tiến trình, những ứng dụng hoặc CSHT ko đc ủng hộ bởi những lời phát ngôn nữa. Mục đích chính của BPMN là cung cấp 1 ký hiệu đồng dạng cho tiến trình mô hình hóa cv trong những kỳ hoạt động và trong những mối qh của chúng.

Hiện nay, BPMN chỉ từ chối duy nhất 1 cú pháp, i.e., 1 ký hiệu (đồ họa) đồng dạng cho những khái niệm mô hình hóa tiến trình cv. Tuy nhiên, có 1 bản đồ chính thức cho XML, được dựa trên việc thi hành tiến trình cv ngôn ngữ BPEL4WS. 1 mô hình hóa meta chính thức ko (chưa) tồn tại.



1. Example model in BPMN

**Testbed** –

Testbed là 1 lời phát ngôn về mô hình hóa cv và về pp đầu tiên đc phát triển bởi tổ chức đào tạo cơ học từ xa cùng với nhiều cty đc liên kết lại. Nó đc chú ý đến vì tiến trình cv và mô hình hóa tổ chức và mục tiêu người SD của nó chủ yếu là những chuyên viên kinh doanh; do đó, lời phát ngôn thiếu sự hoàn hảo về kiến trúc của hệ thống thông tin và những quan niệm có liên quan đến điều này. Bây giờ Cty thiết kế BiZZ bán phần mềm Testbed và pp và bán bộ công cụ hiện đại của cty thiết kế BiZZ (trước kia là Testbed Studio). Testbed hiện giờ đc SD với 1 số lượng cty và viện Chính phủ Hà Lan khổng lồ.

Testbed ghi nhận 3 lĩnh vực sau:

Lĩnh vực actor, mà mô tả từ nguồn gốc của việc thực hiện những hđ kinh doanh;

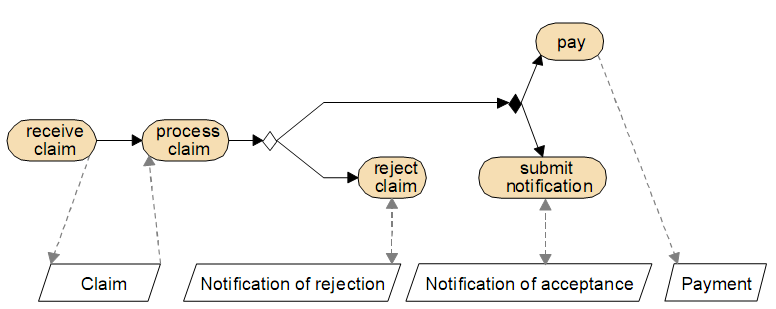
Lĩnh vực behaviour, mà mô tả từ việc thực hiện tiến trình cv bởi nguồn lực;

Lĩnh vực iterm, mà mô tả những object dữ liệu đc điều khiển bởi những tiến trình cv.

3 lĩnh vực trong Testbed cũng có thể đc xem như những loại hình quan điểm đặc biệt. Để chú ý rằng 1 mô hình hoàn chỉnh luôn2 bao hàm sự tượng trưng của tất cả những lĩnh vực này thì ko quan trọng lắm. Hơn nữa những phần đại diện này ko bị cô lập từ những cái khác và chúng tương tác thông qua nhiều cơ chế.

Testbed là 1 ngôn ngữ đồ họa. Hình 2.10 cho ta 1 vd về cách vận hành của mô hình. Nó có 1 sự mô tả hình thức của mô hình meta. Mục tiêu của mô hình meta là cung cấp 1 sự tượng trưng cho cú pháp ngôn ngữ. 1 phần từ những cái này, mô hình tiến trình đc rót vốn với 1 con số có thể có hiệu lực về ngữ nghĩa, có những mục tiêu khó như sự phỏng theo những bậc thang, kiểm tra những mô hình, và những sự phân tích chất lượng.

1 sp2 phụ của Testbed là ngôn ngữ NEML ( ngôn ngữ mô hình hóa táo bạo của hệ thống Network), mà tập trung trong tiến trình kinh doanh điện tử thuộc về tổ chức trong hệ thống liên kết các tổ chức. Thêm vào đó là những lĩnh vực của Testbed, NEML đồng tình ủng hộ những *function* và *role* những lĩnh vực và từ chối 1 số lượng quan niệm phụ thêm vào như *flow* và *transfer.*



1. Example of a business process model in Testbed.

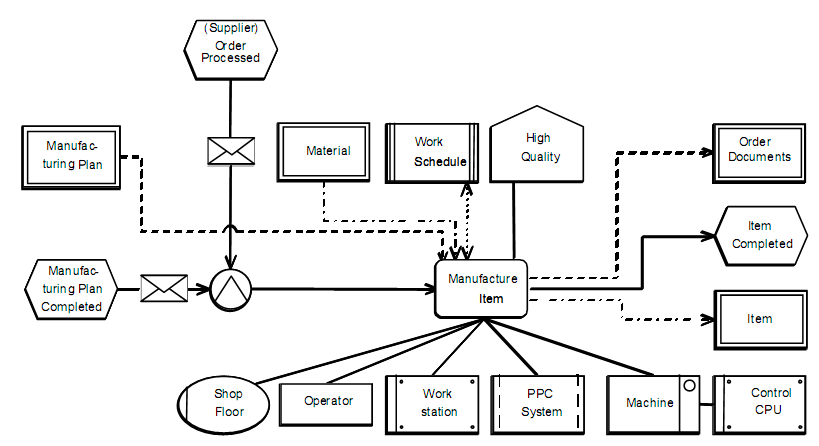
**ARIS (‘Architecture of Integrated Information Systems’, Scheer 1994)** –

ARIS là 1 pp nổi tiếng để tổ chức kinh doanh những mô hình. Mặc dù ARIS đã bắt đầu như 1 nghiên cứu hàn lâm của GS. A.W Scheer, nó có 1 nền tảng CN rõ ràng. Nó ko phải là 1 tiêu chuẩn nhưng nó đc bán rất chạy và do đó nó đc biết rộng rãi. Thêm vào đó là framework thuộc về kiến trúc bậc cao, ARIS là 1 pp mô hình hóa kinh doanh, mà nó đc ủng hộ bởi 1 công cụ phần mềm. IRIS đc nhắm đến để phục vụ cho những mục đích khác nhau: việc cung cấp tài liệu của loại tiến trình cv đang tồn tại, lên kế hoạch cho việc phân tích và thiết kế những tiến trình cv và tán thành việc thiết kế những hệ thống thông tin. Công cụ đc hướng đến cho những người thiết kế hệ thống.

Đề mô hình hóa tiến trình cv trong 1 mô hình tổ chức kinh doanh, ARIS cung cấp 1 ngôn ngữ mô hình đc biết đến như 1 dây chuyền tiến trình dựa theo sự biến. ECP là 1 đồ họa đc sắp xếp trật tự của những sự kiện và những chức năng. Nó cung cấp những bộ liên kết khác nhan mà cho phép luân phiên nhau và song song nhau trong sự thi hành những tiến trình. Hình 2.11 cho ta 1 vd mô hình tiến trình cv đc tạo ra bời ARIS và cũng có sự hiện diện những lời chú dẫn đồ họa đc SD trong những mô hình nà. Quan niệm chính đc xác định bởi ARIS là: những sự kiện, chức năng, sự kiểm soát lưu lượng, mổ xẻ hợp lý, đơn vị những tổ chức, tác động qua lại, đầu ra những dòng chảy, dữ liệu thuộc về môi trường, đầu ra, đầu ra con người, thông tin, mục đích, máy móc, phần cứng máy tính, phần mềm ứng dụng.

Bộ công cụ ARIS bao gồm những sự biên tập khác nhau mà có thể SD để thiết kế và biên tập nhiều loại biểu đồ. Phần quan trọng nhất là những biểu đồ mắc xích giá trị đc thêm vào, những đồ thị, những biểu đồ có tác động qua lại, cây chức năng và EPCs. Trong khi có những lời định nghĩa mang tính hình thức về cú pháp của EPCs, chúng thiếu định nghĩa chính xác về ngữ nghĩa. Nghĩa của EPCs đc đưa ra 1 cách phỏng chừng trong lần xuất bản đầu tiên bởi Scheer. 1 sự thảo luận bao hàm toàn diện về nhược điểm của EPCs có thể đc tìm thấy trong Rittgen. Cũng có trường hợp cho những mô hình oject tương ứng mà đc chuyên môn hóa trong 1 mô hình meta bước đầu. Theo lý do này, ARIS thiếu 1 nền tảng rắn chắc và đc giới hạn SD cho việc thiết kế thuật kiến trúc.

Ký hiệu đồ họa của ARIS thì ko rõ ràng, nhưng khá bao quát, với khá nhiều chỗ quanh co. Trong khi ARIS cho phép những viễn cảnh khác nhau trong tổ chức kinh doanh (quan điểm về dữ liệu, quan điểm về sự kiểm soát, quan điểm về tiến trình/chức năng, quan điểm về cách tổ chức), phép phân tích những khía cạnh này là 1 cái gì đó ko đầy đủ. Do đó, công cụ ko bảo đảm toàn bộ tính toàn vẹn trong những mô hình có tính tương quan với nhau. Tính tích hợp của ARIS đc giới hạn đến những mô hình kinh doanh, và chính xác hơn đến những mô hình thuộc về tổ chức, chức năng và tiến trình. ARIS thì ko bao quát lắm.



1. Events, functions and control flows in ARIS.

**Unified Modeling Language (UML)** –

Ngôn ngữ mô hình hợp nhất (UML) hiện tại là ngôn ngữ tiêu chuẩn CN quan trọng nhất cho những giả tưởng chỉ rõ, tưởng tượng, xây dựng, và chứng minh bằng văn bản của hệ thống phần mềm. Sự phát triển của ngôn ngữ đc quản lý bởi Nhóm quản lý Object. Nó đc nổi bật lên từ việc kết hợp 3 lời chú dẫn hiện nay, Booch, OMT, và Objectory, đc tạo ra bởi “3 người bạn” Booch, Rumbaugh, and Jacobson. Những ảnh hưởng khác từ đồ thị trạng thái của Harel và những chu trình Object của Shlaer-Mellor.

UML hướng đến đc SD bởi những người thiết kế hệ thống. Bởi thế, mô hình UML chỉ rõ ràng đối với những người mà có 1 nền tảng vững vàng trong khoa học máy tính, đặc biệt trong việc định hướng Object. Tuy nhiên, càng bỏ sót những chi tiết kỹ thuật, những mô hình UML nên có sự hiểu biết đầy đủ cho những mục đích minh họa và giải thích cho giới kỹ sư kinh doanh và những nhà chuyên môn. Mặc dù UML đầu tiên đc phát triển cho thiết kế phần mềm Oject đc định hướng, tính hữu ích của nó đc mở rộng đến những khu vực khác, bao gồm mô hình kiến trúc. Tuy nhiên, phiên bản ngôn ngữ đc SD rộng rãi nhất , v1.4 thiếu sự đồng tình ủng hộ của nhiều quan niệm thuộc về kiến trúc. Trong những phiên bản gần đây nhất, UML 2.0, nhiều quan niệm thuộc về kiến trúc đc thêm vào, nhưng phiên bản mới này chưa đc công chúng ủng hộ.

Thông qua việc định hướng object, UML bao trùm tất cả những phạm vi mô hình, người ta nghĩ như thế. Từ cách nhìn của thế giới UML bao gồm chỉ 1 loại cấu thành giống vật thể, đc gọi là Object, cùng với sự liên kết giống như vật, gọi là link. Những vd về Object như cá người, đơn vị tổ chức, sản phẩm, đề án, thành tựu và máy móc. Những Object bao gồm phần tĩnh và phần động. Phần động là sự mô tả cách 1 Object làm việc.

Link phản ánh bất cứ sự kết nối hoặc mqh nào giữa những Oject, sự thay đổi từ cụ thể đến trừu tượng. Do vậy, link có thể biểu thị mqh, sự kiến nối, sự độc lập, sự thích hợp của 1 vật chất, logic, thời gian, cấu trúc, cách cư xử, tương tự nhau hoặc đặc tính bổ sung.

UML là 1 sự kết hợp đầy đủ của 13 ngôn ngữ (sub) mà mỗi cái có 1 mục đích riêng của mục tiêu hoàn chỉnh UML, và mỗi cái có biểu đồ riêng của nó đối với mô hình 1 khía cạnh đặc biệt của hệ thống. 13 biểu đồ có thể đc nhóm lại thành 3 loại:

- Cấu trúc: biểu đồ gói, biểu đồ lớp, biểu đồ hình thể, biểu đồ cấu trúc đa hợp

- Cách cư xử: SD những biểu đồ trường hợp, biểu đồ trạng thái, biểu đồ nối tiếp, biểu đồ tính toán thời gian, biểu đồ giao tiếp, biểu đồ hoạt động, biểu đồ khái quát tác động qua lại.

- Sự thực thi: biểu đồ hợp thành, biểu đồ triển khai.

Mỗi loại biểu đồ mô tả 1 hệ thống hoặc 1 phần của nó từ 1 quan điểm chắc chắc, và bao gồm những ký hiệu tượng trưng riêng của nó. Tuy nhiên, những loại biểu đồ này và mô hình meta UML thì có quan hệ với nhau; ko có sự chia cắt rõ ràng giữa cách nhìn quan niệm mô hình meta đã đc tạo ra. Do đó, mqh giữa những quan niệm mô hình hóa trong những biểu đồ khác nhau thì thường ko rõ ràng. Chúng ta sẽ ko hiển thị ký hiệu của tất cả những biểu đồ này và những quan niệm mô hình hóa ở đây, 1 cách nhìn đúng đắn đc đưa ra trong Fowler & Scott.

Hơn nữa, 1 phần từ biểu đồ gói, biểu đồ thành phần, và biểu đồ triển khai, mỗi cách nói khác nhau thì tự bản thân nó có sự kết hợp về 1 dãy building khác nhau. 1 số loại ngôn ngữ này có sự chồng xếp lên nhau lớn…biểu đồ hoạt động và biểu đồ trạng thái. Thuận lợi của sự phong phú là tính cảm xúc của ngôn ngữ; sự bất lợi là tính dễ đọc và dễ bị ảnh hưởng của ngôn ngữ. 1 lượng lớn ký hiệu tượng trưng và biểu đồ tạo nên đường cong của UML bước hoàn thiện cho người SD mới.

Kế đến là ký hiệu đồ họa, UML bao gồm ngôn ngữ ràng buộc Object (OCL), 1 ngôn ngữ nguyên bản cho những yếu tố mô hình ràng buộc chuyên môn hóa. Nghĩa của biểu đồ UML luôn luôn ko trực giác và đôi lúc đòi hỏi phải tìm hiểu cẩn thận. 1 kinh nghiệm của người SD UML, tuy nhiên, việc SD ngôn ngữ thì ko quá khó.

Để mở rộng từ ngữ mô hình hóa hoặc đưa ra con dấu đặc biệt thì hầu như ám chỉ đến loại hình trừu tượng chắc chắn thường xuất hiện, UML đưa ra 3 loại cơ chế:

- 1 “bản mẫu” là 1 sự mở rộng của vốn từ ngữ UML mà cho phép tạo ra nhiều dãy building mới, dựa trên những cái đã có sẵn. 1 bản mẫu đc SD để làm rõ sự chuyên môn hóa của những yếu tố tồn tại của mô hình meta UML.

- 1 “giá trị buộc” là sự mở rộng những thuộc tính của 1 yếu tố UML cho phép tạo ra thông tin mới trong sự chuyên môn hóa của yếu tố đó. Giá trị buộc có thể đc thêm đối với tất cả những yếu tố mô hình meta đang tồn tại.

- UML đưa ra khả năng để làm rõ cái gọi là “ profiles” đc làm hòa hợp đến phạm vi vấn đề chắc chắn. 1 profile là 1 loại hình thái ngôn ngữ của ngôn ngữ mô hình gốc, đc làm phù hợp hơn để phản ảnh những ký tự của 1 vấn đề chắc chắn. 1 profile SD giá trị buộc và bản mẫu để hiển thị 1 mô hình đặc biệt và chính xác.

1 profile mà là sự thích hợp đặc biệt cho kiến trúc khó làm thì là profile cho EDOC. Mục đích của nó là cung cấp cấu trúc và mô hình hóa cho việc cộng tác hoặc Internet

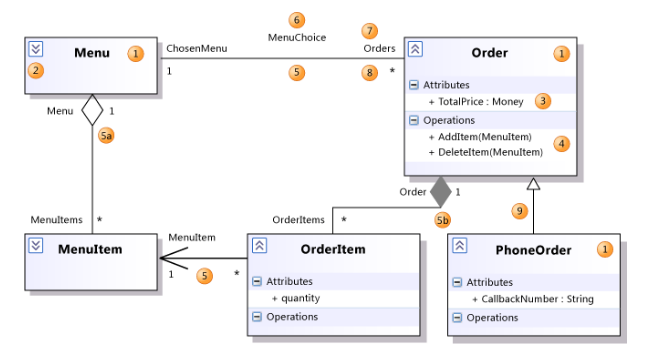
Computing, với những kỹ thuật như Web services, Enterprise Java Beans, và soạn Corba. Profile EDOC đc chọn bởi 1 tiêu chuẩn vào 11/2001 và sẽ cung cấp sự phát triển mô hình chạy đua của hệ thống enterprise đc dựa trên kiến trúc mô hình chạy đua. Profile EDOC cung cấp 1 kiến trúc cộng tác kinh doanh, 1 kiến trúc hợp thành kinh doanh kỹ thuật độc lập, và những ý niệm mô hình cho việc mô tả những tiến trình kinh doanh, những ứng dụng và CSHT.

Mặc dù những cơ chế mở rộng này đưa ra tính linh hoạt lớn của UML, chúng cũng có những điểm yếu của ngôn ngữ. Đặc biệt là bản mẫu, khi ứng dụng quá nhiều, có thể làm lúng túng người đọc, người ta ko quen với chúng. Trong trường hợp bản mẫu mang về những điểm mạnh của UML, chúng đạt đc sự tiêu chuẩn hóa.

Một phần của UML có nền tảng thuộc bản chất. Điều chúng đang bỏ lỡ là ngữ nghĩa chính và tính nhất quán thường xuyên. Vì đây là 1 chủ đề của việc nghiên cứu những điều xảy ra và quan sát đc, chúng có thể mong đợi sự cải tiến thực tế những gì liên quan đến ngữ nghĩa chính và tính nhất quán thường xuyên trong tương lai gần đây. Nghĩa cho những loại biểu đồ cá nhân thì tồn tại, trong nhiều hoặc ít theo lối hình thức. Tuy nhiên, 1 ngữ nghĩa thống nhất thì vẫn thiếu.

Việc thiếu 1 ngữ nghĩa thống nhất tạo ra sự khó khăn để cung cấp sự đồng tình theo lôi phân tích của UML. Việc phân tích đc giới hạn đến cái mà đc cho phép trong biểu đồ đơn tính và từ khi nghĩa của UML ko đc chuyên môn hóa tốt theo tiêu chuẩn, kỹ thuật phân tích chặt chẽ thì khó xác định.

Có lẽ phần quan trọng nhất của UML là bộ công cụ mở rộng của nó: có nhiều thương nghiệp cũng như nhiều môi trường mô hình trong lĩnh vực công cộng. Vì UML khá lớn, hầu hết chúng ko (chưa) phủ lắp mọi thứ, và sự đồng tình ủng hộ đặc biệt cho phiên bản mới nhất gần đây, UML 2.0, nó vẫn còn thiếu. Vì nhiều môi trường cung cấp nghĩa để phiên dịch 1 mô hình thành code có thể thực hiện đc,…Java, 1 vài dạng của sự phân tích đang đc cung cấp: thông qua việc thực thi. Thường cũng có nhiều nghĩa phân tích và xác minh khác đang đc cung cấp, qua việc kiểm tra tính nhất quán từng bộ phận hoặc những dạng của tính linh hoạt hoặc dứt khoát đối với 1 lĩnh vực khác nơi mà việc xác minh rõ ràng có thể đc thực hiện.



1. Ví dụ một mô hình UML

**‘Architecture Description Language’ (ADL)** –

Thuật ngữ “Ngôn ngữ mô tả thuộc về kiến trúc” (ADL) đc SD để chỉ dẫn đến 1 loại ngôn ngữ để miêu tả 1 kiến trúc phần mềm trong thuật ngữ nói chung. Trạng thái phổ biến hiện có của ADL, với nhiều điều khác trong những quan niệm đúng đắn rằng chúng thừa nhận: 1 vài điểm chủ yếu trong phương diện cấu trúc của kiến trúc XD, trong khi những cái khác chú ý hơn tới khía cạnh động lực. Nói chung, những quan điểm của chúng đc xác định rõ ở mức độ tổng quát: mặc dù chúng thường đc dùng cho cấp độ ứng dụng mô hình, cái lợi của những quan niệm này là ko bị giới hạn đối với cái khác. Theo kết quả của mức độ trừu tượng này, việc XD và tìm hiểu đặc điểm kỹ thuật của ADL có thể gây khó khăn cho những người SD ko phải là chuyên gia. 1 thuận lợi là việc xác định chính xác và là nển tảng thuộc về bản chất của ngôn ngữ, mà có thể khiến chúng trở nên phù hợp hơn như 1 ngôn ngữ cơ sở cho những quan điểm đặc biệt hơn. Theo Medvidovic and Taylor, căn bản của ADL là sự mô tả, và số lượng lớn ADL đc so sánh.

Mặc dù những quan niệm đc SD trong ADL thì tổng quát nhưng chúng đc SD chủ yếu trong lĩnh vực kiến trúc phần mềm. Điều thêm nữa cho ADL với khả năng tổng quát có thể ứng dụng đc, có ADLs với nhiều lĩnh vực ứng dụng đặc biệt hơn. Vì mức độ tập trung cao và có bản chất hình thức của những khái niệm, ADL có tính phù hợp thiết yếu cho người SD với 1 nền tảng kỹ thuật. Chúng ko phù hợp như 1 phương tiện giao tiếp ở cấp độ tổ chức.

Theo nguyên tắc, quan niệm ADL thì khá linh hoạt để tạo ra những mô hình trong 1 vài lĩnh vực. Tuy nhiên, chúng đc ứng dụng chính, và phù hợp nhất cho lĩnh vực ứng dụng. Khi Acme đc tuyên bố là phù hợp như là 1 sự mô tả thuộc về kiến trúc tổng quát và ngôn ngữ có thể thay thế qua lại thì chúng ta tin rằng những quan điểm của nó có thể đc SD như là 1 sự khiển trách cho ADL. Quan điểm cốt lõi là:

- Tính hợp thành

- Bộ liên kết

- Hệ thống

- Cổng

- Vai trò

- Sự tượng trưng

- Bản đồ phụ

ADL nói chung giống Acme đều có 1 nền tảng thuộc hàn lâm, và đc giới hạn về cách SD. Tuy nhiên, 1 vài quan niệm này hiện nay đc bao gồm trong UML 2.0. Do đó, những quan niệm này đc tạo ra có cơ sở SD lớn và sẽ đc đồng tình bởi 1 phạm vi rộng lớn của bộ công cụ phần mềm.

**Suitability for Enterprise Architecture** –

Trong những phần trước, chúng ta đã có 1 cách nhìn tổng thể về ngôn ngữ hiện tại cho việc mô hình hóa những khu vực tổ chức, tiến trình kinh doanh, trình ứng dụng, và kỹ thuật. Rõ ràng rằng ko có những điều này đã tạo nên thành công trong việc trở thành “ ngôn ngữ” mà có thể bao trùm mọi lĩnh vực. Chung quy, có 1 số phương diện mà hầu như tất cả những ngôn ngữ đạt ở mức độ thấp:

- Mối tương quan giữa những lĩnh vực thì khó xác định, và những mô hình đc tạo ra trong cách nhìn khác nhau thì ko đc hợp nhất lắm.

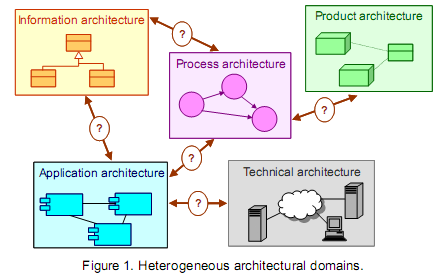
- Hầu như các ngôn ngữ có 1 nền tảng về hình thức kém và về ngữ nghĩa thì thiếu sự xác định rõ ràng.

- Hầu như các ngôn ngữ thiếu sự nhìn nhận toàn bộ về mặt kiến trúc và bị hạn chế đối với việc kinh doanh cũng như lĩnh vực phụ về trình ứng dụng và kỹ thuật.

Tương phản với mô hình tiến trình tổ chức và kinh doanh, ko có ngôn ngữ thống trị nào, trong mô hình hóa trình ứng dụng và kỹ thuật UML đã trở thành 1 tiêu chuẩn thế giới đúng đắn. UML là phương pháp mô hình chính cùng với ICT, và lợi ích của nó đang đc lan rộng đến nhiều khu vực khác. Điều này khiến UML trở thành 1 ngôn ngữ quan trọng ko chỉ cho hệ thống phần mềm mô hình hóa mà còn cho tiến trình kinh doanh và lĩnh vực kiến trúc nói chung. Tuy nhiên, UML ko dễ gì bị ảnh hưởng và có thể hiểu đc đối với các chuyên viên quản lý và kinh doanh; do đó, cần cung cung cách hình dung đặc biệt và xem xét mô hình UML. Đưa ra tầm quan trọng của UML, những ngôn ngữ mô hình hóa khác sẽ đc cung cấp giống như bề mặt chung v2 sự sắp đặt đv chúng.

* 1. **Archimate language**
     1. **ArchiMate là gì**

ArchiMate là một ngôn ngữ dùng để miêu tả mô hình kiến trúc enterprise , hỗ trợ các 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. Heterogenneou sadadasd

Cách tiếp cận việc tích hợp, thiết kế và xây dựng mô hình Business và hệ thống CNTT là không thể thiếu trong các công ty hiện nay . Cũng giống như một bản vẽ kiến trúc  
xây dựng, kiến trúc cổ điển mô tả các khía cạnh khác nhau của việc xây dựng một hệ thống . ArchiMate cung cấp một ngôn ngữ chung để mô tả và phân tích xây dựng hoạt động của các quá trình Business, cơ cấu tổ chức, dòng chảy thông tin, hệ thống CNTT, và cơ sở hạ tầng kỹ thuật và sự phụ thuộc đa phía của họ. Điều này giúp các bên liên quan thiết kế, đánh giá, và truyền đạt thông tin cần trao đổi một cách dễ dàng hơn .

* + 1. **Tại sao dùng ArchiMate**

Trong nhiều tổ chức công ty công nghệ thông tin chuyên sâu, đã tồn tại những loại hình kiến trúc và các kiến trúc sư . Sự khác biệt giữa kiến trúc và thiết kế không phải luôn luôn sắc nét , cả hai đều mô tả các mối quan hệ giữa nhiều ứng dụng phần mềm được sử dụng trong doanh nghiệp, cũng như cấu trúc toàn cầu nội bộ của các ứng dụng này.Hiện nay, UML thường là ngôn ngữ của sự lựa chọn cho mục tiêu này, mặc dù vẫn còn có những tổ chức sử dụng ký hiệu riêng độc quyền của họ.

Trong nhiều ngành kinh doanh theo định hướng ", làm việc theo kiến trúc" gần đây được phát triển. Kể từ sự ra đời của quá trình định hướng trong những năm 1990,ngày càng nhiều tổ chức đã bắt đầu tài liệu các quy trình kinh doanh của mình trong một nhiều hơn hay ít hơn cách chính thức. Tuy nhiên, những mô tả này không tập trung vào các khía cạnh kiến trúc, nghĩa là, họ không cung cấp một tổng quan về cấu trúc toàn cầu trong các quy trình và các mối quan hệ giữa chúng. Một số tổ chức có một mô tả của danh mục sản phẩm của họ, mà thường là dựa trên văn bản: hình ảnh mẫu chưa được chấp nhận trong lĩnh vực này.

Các tổ chức cần phải thích ứng ngày càng nhanh chóng và thay đổi dự kiến của yêu cầu khách hàng và mục tiêu kinh doanh. Nhu cầu này ảnh hưởng đến toàn bộ hệ thống hoạt động của một enterprise, từ cơ cấu tổ chức để các cơ sở hạ tầng mạng. Làm thế nào có thể bạn kiểm soát các tác động của những thay đổi này? Kiến trúc có thể là câu trả lời. Các dự án ArchiMate sẽ phát triển một phương pháp tiếp cận tích hợp kiến trúc mô tả và hình dung các miền enterprise khác nhau và quan hệ của chúng .Sử dụng những kiến trúc tích hợp viện trợ các bên liên quan trong việc đánh giá tác động của sự lựa chọn thiết kế và thay đổi. Kiến trúc là một tổng thể nhất quán của các nguyên tắc, phương pháp và mô hình được sử dụng trong thiết kế và thực hiện cơ cấu tổ chức, quy trình kinh doanh, hệ thống thông tin và cơ sở hạ tầng. Tuy nhiên, các lĩnh vực này không được tiếp cận một cách tích hợp, mà làm cho nó khó khăn để đánh giá những tác động của thay đổi đề xuất.Mỗi miền nói ngôn ngữ riêng của mình, rút ra các mô hình riêng của mình, và sử dụng các kỹ thuật riêng và các công cụ của nó. 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 với các công cụ hỗ trợ và cải thiện quá trình kiến trúc hiện có và tiêu chuẩn mới nổi sẽ được sử dụng hoặc tích hợp bất cứ khi nào có thể. ArchiMate sẽ tích cực tham gia các diễn đàn quốc gia và quốc tế và các tổ chức tiêu chuẩn hóa, để thúc đẩy việc phổ biến kết quả dự án.  
  
Kiến trúc cung cấp một phương tiện để xử lý sự phức tạp của hiện đại, thông tin thiếu khả dụng nhiều doanh nghiệp. Để thực hiện được điều này, kiến trúc sư cần những cách để thể hiện kiến trúc như một cách rõ ràng nhất có thể: cả hai cho sự hiểu biết của mình và để liên lạc với các bên liên quan khác, chẳng hạn như phát triển hệ thống,người dùng cuối, và quản lý. Thật không may, tình hình hiện nay là Architects trong các lĩnh vực khác nhau, thậm chí trong cùng một tổ chức, thường sử dụng riêng các kỹ thuật mô tả và công ước của họ. Đến nay, không có tiêu chuẩn ngôn ngữ để mô tả kiến trúc doanh nghiệp một cách chính xác qua biên giới tên miền. 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) được khó hiểu cho người không chuyên gia. Điều này thường xuyên dẫn đến hiểu lầm làm cản trở sự hợp tác của kiến trúc sư và các bên liên quan khác. Ngoài ra, nó làm cho nó rất khó khăn để cung cấp các công cụ cho hình dung và phân tích của các kiến trúc.

Kiến trúc enterprise là một công cụ quan trọng để hội nhập toàn công ty. Đây là một toàn bộ mạch lạc của nguyên tắc, phương pháp và mô hình được sử dụng trong việc thiết kế và thực hiện cơ cấu tổ chức của doanh nghiệp, quy trình kinh doanh, hệ thống thông tin và cơ sở hạ tầng CNTT. Một hành nghề kiến trúc tốt cho phép một tổ chức để sắp xếp các hoạt động kinh doanh và CNTT với chiến lược của mình, nhanh chóng đáp ứng với những thay đổi trong môi trường, và sử dụng tối ưu các cơ hội công nghệ. Sự phát triển và bảo trì các kiến trúc sẽ dẫn đến hiệu quả, giảm chi phí, và tính linh hoạt.Trong thời hạn các công ty kiến trúc miền khác nhau có thể được tìm thấy, như tổ chức, quy trình kinh doanh, ứng dụng, thông tin, và kiến trúc kỹ thuật. Mỗi miền kiến trúc có khái niệm riêng của mình cho các mô hình và trực quan của sự gắn kết nội bộ của nó. Những mô hình cụ thể và đơn giản hóa sự hình dung việc giao tiếp, thảo luận và phân tích trong miền .Tuy nhiên, quan hệ giữa các khái niệm trong các miền khác nhau trong nhiều trường hợp không rõ ràng. Hơn nữa, các miền này thường một phần chồng lên nhau nhưng sử dụng khái niệm khác nhau để thể hiện những ý tưởng tương tự .Kết quả là sự mơ hồ và đứng nhầm lẫn trong cách của các tổ chức cách linh hoạt và hiệu quả hoạt động .

ArchiMate muốn làm giảm đi những sự mơ hồ đó. Nó thể hiện một cách thống nhất của mô hình kiến trúc doanh nghiệp, tích hợp các lĩnh vực khác nhau và mô tả chúng một cách dễ dàng đọc được ArchiMate là tất nhiên không phải là một phát triển cô lập. Các mối quan hệ với các phương pháp hiện hành và kỹ thuật, giống như ngôn ngữ mô hình hóa như UML và BPMN, và các phương pháp và khuôn khổ như TOGAF và Zachman, đều được mô tả.

Vì vậy, chúng tôi có thể nói rằng trong nhiều lĩnh vực chuyên môn khác nhau được trình bày trong một doanh nghiệp, một số loại thực hành kiến trúc tồn tại, với mức độ khác nhau của sự thuần thục. Tuy nhiên, do tính không đồng nhất của những phương pháp và kỹ thuật được sử dụng để tài liệu kiến trúc, nó là rất khó để xác định các lĩnh vực khác nhau được nhau. Tuy nhiên, rõ ràng là có phụ thuộc mạnh mẽ giữa các lĩnh vực. Ví dụ: mục tiêu của doanh nghiệp (chính) quy trình của một tổ chức là để nhận ra sản phẩm của mình; ứng dụng phần mềm hỗ trợ quy trình kinh doanh, trong khi cơ sở hạ tầng kỹ thuật là cần thiết để chạy các ứng dụng, thông tin được sử dụng trong các quá trình kinh doanh và xử lý bởi các ứng dụng . Đối với giao tiếp tối ưu giữa các kiến trúc sư miền, cần thiết để canh chỉnh thiết kế trong các lĩnh vực khác nhau, một hình ảnh rõ ràng về sự liên thuộc miền là không thể thiếu.  
  
Với những quan sát này trong tâm trí, người ta kết luận rằng một ngôn ngữ để mô hình hóa doanh nghiệp kiến trúc nên tập trung vào quan hệ liên miền. Với ngôn ngữ như vậy, chúng tôi phải được có khả năng mô hình :

Cơ cấu toàn cầu trong miền, thể hiện các yếu tố chính và của họ   
phụ thuộc, theo một cách dễ hiểu cho người không chuyên gia của các tên miền.

Các mối quan hệ giữa các miền

Một đặc tính quan trọng của một ngôn ngữ mô hình doanh nghiệp - như đối với bất kỳ mô hình   
ngôn ngữ - là một nền tảng chính thức, trong đó đảm bảo rằng các mô hình có thể được giải thích trong một cách rõ ràng và họ đều tuân theo phân tích tự động. Ngoài ra, cần có thể hình dung mô hình theo một cách khác nhau, phù hợp đối với các bên liên quan cụ thể với các yêu cầu thông tin cụ thể. Không có trong các ngôn ngữ hiện đang tồn tại mô hình hoàn toàn đáp ứng các yêu cầu này. Mặc dù, về nguyên tắc, những khái niệm của ngôn ngữ ArchiMate này là đầy đủ và chung chung ý nghĩa để mô hình nhiều khía cạnh trong lĩnh vực khác nhau, đó rõ ràng là không phải ý định của chúng tôi để giới thiệu một ngôn ngữ có thể thay thế tất cả các ngôn ngữ miền cụ thể mà tồn tại. Đối với cụ thể (chi tiết) thiết kế, ví dụ như, quy trình kinh doanh hoặc các ứng dụng, ngôn ngữ hiện có (UML) có thể sẽ là thích hợp hơn. Tuy nhiên , trong ngôn ngữ mà chúng ta nói đến sẽ phù hợp nhiều nhất có thể mô hình những chuẩn mà tồn tại trong những miền khác nhau .

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

1. Nó là một tiêu chuẩn nhà cung cấp độc lập Quốc tế của 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.
2. Nó 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 mờ" về kiến trúc.
3. Nó là một ngôn ngữ vừa và đơn giản. Nó chứa chỉ đủ cho khái niệm mô hình kiến trúc enterprise và không cồng kềnh để bao gồm mọi thứ có thể . Cấu trúc đồng nhất của nó làm cho nó dễ dàng để học hỏi và áp dụng.
4. Nó 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 được vay mượn từ các ngôn ngữ khác như UML hoặc BPMN, để cung cấp cầu nối giữa chúng dễ dàng.
5. Nó không quy định một cách để làm việc, nhưng nó là cách dễ dàng kết hợp với phương pháp hiện có như TOGAF.
6. Nó đã được thử và thử nghiệm của các tổ chức người sử dụng khác nhau và được hỗ trợ bởi nhiều tư vấn và các công cụ phần mềm.
   * 1. **Các khái niệm chính trong ngôn ngữ**

Trong ngôn ngữ mô hình doanh nghiệp mà chúng tôi đề xuất, khái niệm dịch vụ đóng một trung tâm vai trò. dịch vụ được định nghĩa là một đơn vị chức năng rằng một số thực thể (ví dụ, một hệ thống, tổ chức, bộ phận) làm cho có sẵn cho môi trường của nó, và trong đó có một số giá trị cho các chủ thể nhất định trong môi trường. Dịch vụ hỗ trợ định hướng các xu hướng hiện nay như nền kinh tế dựa trên dịch vụ mạng và tích hợp công nghệ thông tin với các dịch vụ Web. Những ví dụ này đã cho thấy rằng dịch vụ của một tính chất rất khác nhau và có thể được nhận thức: chúng có thể được cung cấp bởi các tổ chức cho khách hàng của họ, bởi các ứng dụng để quy trình kinh doanh, hoặc bằng phương tiện công nghệ (ví dụ, mạng lưới thông tin liên lạc) để ứng dụng.

Một lớp cung cấp một cách tự nhiên để nhìn vào mô hình dịch vụ theo định hướng. Các lớp cao hơn sử dụng dịch vụ được cung cấp bởi các lớp thấp hơn. Mặc dù, ở một mức độ trừu tượng, những khái niệm được sử dụng trong mỗi lớp cũng tương tự, chúng ta định nghĩa khái niệm cụ thể hơn được cụ thể cho một lớp nhất định. Trong bối cảnh này, chúng tôi phân biệt ba lớp chính:

• Lớp Business cung cấp sản phẩm và dịch vụ cho khách hàng bên ngoài, đó là

nhận ra trong tổ chức của quy trình business được thực hiện bởi các actor business và roles.

• Lớp Application hỗ trợ các lớp business với các dịch vụ ứng dụng được

nhận ra bởi (phần mềm) ứng dụng.

• Lớp Technology cung cấp dịch vụ cơ sở hạ tầng (ví dụ, xử lý, lưu trữ và

dịch vụ truyền thông) cần thiết để chạy các ứng dụng, thực hiện bởi máy tính và

thông tin liên lạc phần cứng và phần mềm hệ thống.

Mỗi lớp chính có thể được chia tại các lớp. Ví dụ, trong Lớp Business, Business chính quy trình thực hiện các sản phẩm của một công ty có thể làm cho việc sử dụng một lớp trung học (hỗ trợ) quy trình Business; trong lớp ứng dụng, các ứng dụng người dùng cuối có thể sử dụng dịch vụ được cung cấp chung chung bằng cách hỗ trợ ứng dụng. Ngày đầu của lớp Business, một lớp môi trường riêng biệt có thể được thêm, Mô hình hóa các khách hàng bên ngoài mà làm cho việc sử dụng các dịch vụ của tổ chức

(Mặc dù này cũng có thể được coi là một phần của lớp Business). Phù hợp với định hướng dịch vụ, các mối quan hệ quan trọng nhất giữa các lớp được hình thành bởi các mối quan hệ sử dụng, trong đó thể hiện như thế nào các lớp cao hơn sử dụng các dịch vụ của các lớp thấp hơn. Tuy nhiên, một loại thứ hai của liên kết được hình thành bởi những quan hệ thực hiện: các yếu tố trong các lớp thấp hơn có thể nhận ra các yếu tố so sánh trong các lớp cao hơn, ví dụ, dữ liệu một 'đối tượng' (Application layer) có thể nhận ra một đối tượng Business "(Business layer), hoặc một 'artifact của lớp Công nghệ () có thể nhận ra hoặc là một' đối tượng dữ liệu 'hoặc thành phần một ứng dụng "(Application layer).

Cấu trúc chung của các mô hình trong các lớp khác nhau là như nhau. Cùng loại khái niệm và quan hệ được sử dụng, mặc dù bản chất chính xác của họ và độ chi tiết khác biệt. khái niệm hành vi được giao cho các khái niệm cấu trúc, hiển thị những người hoặc những gì hiển thị hành vi. Trong ví dụ, vai trò, giao diện và cộng tác được giao cho quá trình Business, dịch vụ tổ chức và

Business tương tác, tương ứng.

Thứ hai, chúng tôi phân biệt giữa một cái nhìn bên ngoài và xem một nội bộ về hệ thống.

Khi nhìn vào khía cạnh hành vi, quan điểm này phản ánh các nguyên tắc của dịch vụ

định hướng như được giới thiệu trong phần trước. Khái niệm dịch vụ đại diện cho một đơn vị

chức năng thiết yếu mà một hệ thống cho thấy nhiều đến môi trường của nó. Đối với những người sử dụng bên ngoài, chỉ có chức năng này bên ngoài, cùng với các khía cạnh phi chức năng như chất lượng dịch vụ, chi phí, vv, có liên quan. Nếu cần, chúng có thể được quy định tại hợp đồng hoặc thỏa thuận cấp độ dịch vụ. Dịch vụ có thể truy cập thông qua giao diện, mà tạo thành hình dáng bên ngoài vào khía cạnh cơ cấu.

Mặc dù cho người sử dụng bên ngoài chỉ nhìn bên ngoài là có liên quan, việc thiết kế

tổ chức, hệ thống và các hoạt động nội bộ và quản lý cũng đòi hỏi kiến thức về việc thực hiện nội bộ của các dịch vụ và các giao diện. Để thực hiện điều này, chúng tôi phân biệt giữa hành vi đó được thực hiện bởi một yếu tố cấu trúc cá nhân (ví dụ, actor, vai trò thành phần, vv), hoặc hành vi của tập thể (tương tác) được thực hiện bởi một sự hợp tác của nhiều yếu tố cấu trúc.

Ngoài yếu tố tích cực cơ cấu (các actor Business, các thành phần ứng dụng và thiết bị hiển thị hành vi thực tế, nghĩa là, các đối tượng "của hoạt động), chúng tôi cũng nhận ra các yếu tố cấu trúc thụ động, nghĩa là, các đối tượng trên mà hành vi được thực hiện. Trong lĩnh vực tổ chức thông tin chuyên sâu, là trọng tâm chính của ngôn ngữ của chúng tôi, đây là những đối tượng thường là thông tin trong các lớp Business và đối tượng dữ liệu trong lớp ứng dụng, nhưng họ cũng có thể được sử dụng để đại diện cho các đối tượng vật lý.

* + 1. **Những ký hiệu của ngôn ngữ**

RelationShips

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

Structural Relationship

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 hoặc data object |  |
| Used by | Mô hình mối quan hệ sử dụng bởi ví dụ dịch vụ được sử dụng bởi process, hay interface được sử dụng bởi 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

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 |  |

Orther Relateionship

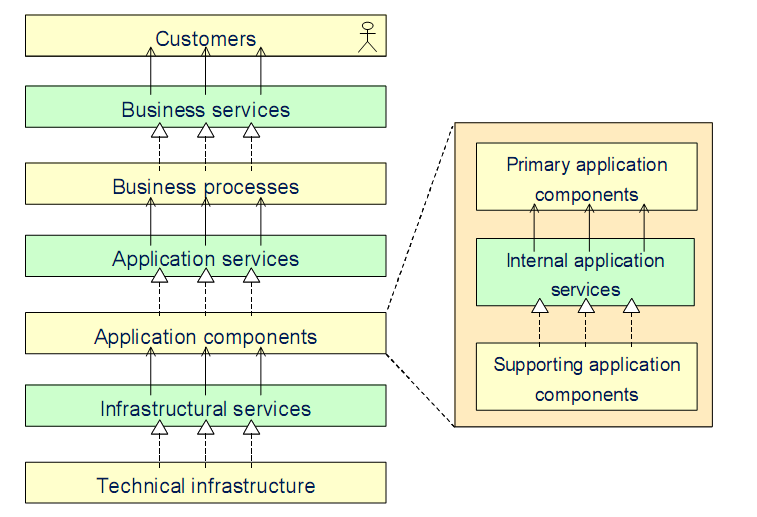
|  |  |  |
| --- | --- | --- |
| Orther Relationship | | Ký hiệu |
| Grouping | Nhóm những đối tượng cùng loại hay khác loại dựa vào một đặc tính chung nào đó |  |
| Junction | Dùng để kết nối những mối quan hệ giống nhau |  |
| Specialization | Thể hiện mối liên kết mà một đối tượng chuyên môn hoá hay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

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

Một xem lớp cung cấp một cách tự nhiên để nhìn vào mô hình dịch vụ theo định hướng. Các lớp cao hơn sử dụng dịch vụ được cung cấp bởi các lớp thấp hơn. ArchiMate phân biệt ba lớp chính:

Lớp Business cung cấp sản phẩm và dịch vụ cho khách hàng bên ngoài, được nhận ra trong tổ chức của quy trình Business được thực hiện bởi các actor Business và vai trò.

Lớp Application hỗ trợ các lớp Business với các dịch vụ ứng dụng được nhận ra bởi (các thành phần phần mềm ứng dụng).



Lớp Technology 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.

To illustrate the use of our modelling language, we introduce the fictitious

(though realistic) insurance company ArchiSurance. ArchiSurance origi-

nally provided home and travel insurance, but merged recently with two

other insurance companies, PRO-FIT (car insurance) and LegallyYours

(legal aid insurance). By streamlining their operations and removing dupli-

cation, substantial synergy is expected from this merger.

ArchiSurance’s management is now wrestling with the intricacies of in-

tegrating these three companies, and has decided to take an enterprise ar-

chitecture approach to create more insight into this complexity. To provide

a high-level overview of ArchiSurance’s primary operations, the company

is described in terms of its main business functions:

− Maintaining Customer Relations and Intermediary Relations: these busi-

ness functions are responsible for the contacts of ArchiSurance with its

customers and the intermediaries that sell its products. This function

handles customer questions and incoming claims, and performs market-

ing and sales.

− Contracting: this function does the ‘back-office’ processing of contracts.

It performs risk analysis and ensures legally and financially correct con-

tracts. Modelling Example 109

− Claims Handling: this function is responsible for handling insurance

claims.

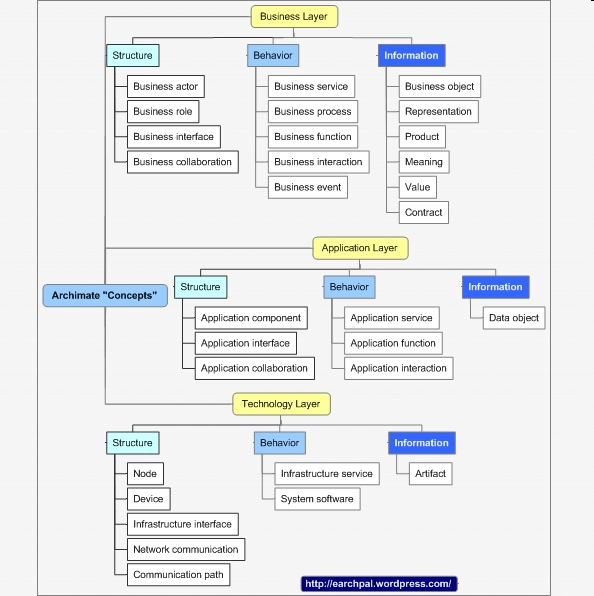
− Financial Handling: this function performs the regular premium collec-

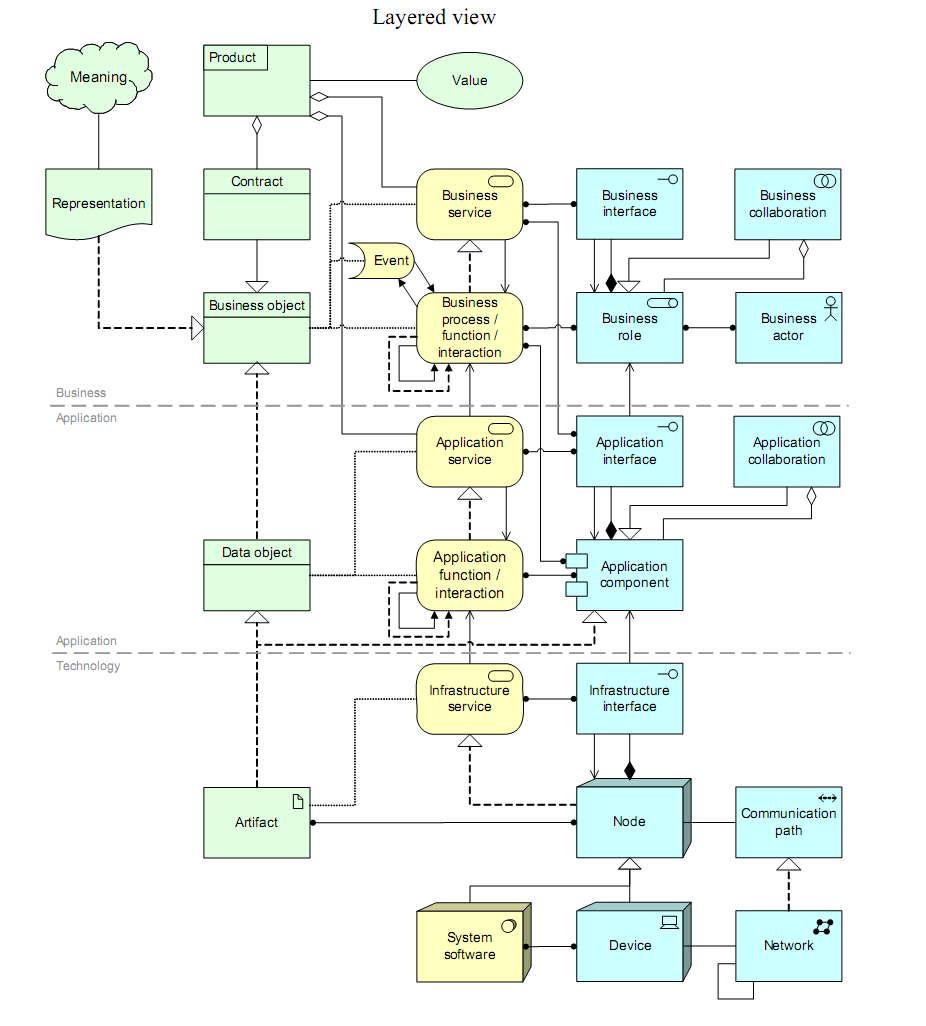
tion, according to the insurance policies with customers as produced by

Contracting, and handles the payment of insurance claims.

− Asset Management: this function manages the financial assets of Archi-

Surance, e.g., by investing in stocks and bonds.

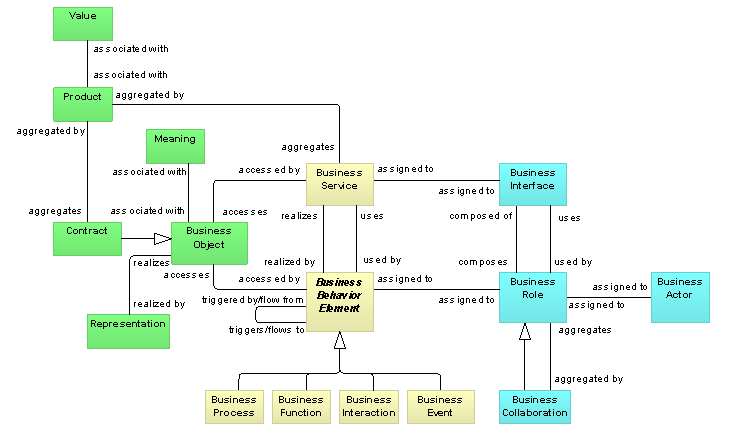




1. Tổng quan về các khái niệm ArchiMate và mối quan hệ chính.
   * 1. **Tầng Business**

Application Layer

Application Meta Model



Ở cấp độ high level các layer trong ngôn ngữ archimate đều có các khía cạnh thể hiện rất giống nhau, vì vậy business layer cũng được phân biệt ở các khía cạnh sau:

Khía cạnh structural:

Khía cạnh active

Khía cạnh passive hay khía cạnh data

Khía cạnh behavioral

Khía cạnh information

Structual Concepts-Khái niệm thuộc về cấu trúc

Khía cạnh active:

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.

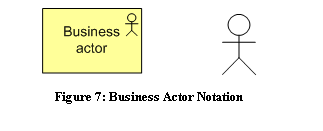
Business Actor

Khía cạnh structure của business layer thường quy đến kết cấu của tổ chức, khái niệm trung tâm của khía cạnh structure 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.

Định nghĩa:

Business actor là một thực thể chủ động(active entity) thực hiện những ứng xử(behaviour)

Kí hiệu minh hoạ:

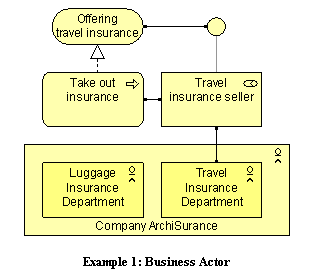


Tổng quát Business Actor :

Một business actor thực hiện những ứng xử được gán với một hay nhiều business role

Tên của một business actor nên được thể hiện là danh từ

Hình minh họa:



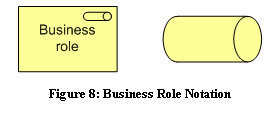
Business Role

Trong business layer, để tăng sự linh động trong mối liên kết giữa những actor và những ứng xử(behaviour), 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 role 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 role để mô tả một tổ chức sẽ giúp tăng tính ổn định hơn nhiều so với việc sử dụng những actor đảm nhiệm những role đó. Thứ hai, một role có thể do nhiều actor đảm nhiệm, và ngược lại, một actor có thể đảm nhiệm nhiều role. Role giúp có thể phân biệt trách nhiệm rõ ràng của từng actor trong tổ chức.

Định nghĩa:

Business Role phát biểu một ứng xử nào đó được thực hiện bởi một business actor đảm nhiệm vai trò(role) đó

Ký hiệu minh họa:



Tổng quát Business Role:

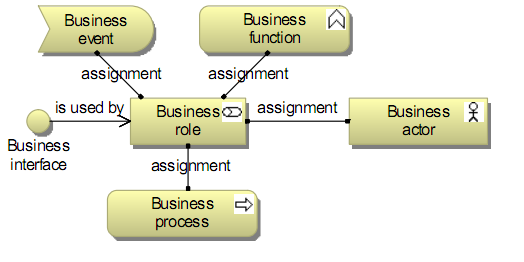
Business Role được gán với một business actor

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 thể hiện là danh từ

Hình minh họa:



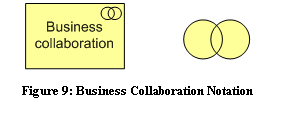
Business Collaboration

Architectural descriptions tập trung chủ yếu vào cấu trúc(structure), 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 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 lá một tập của nhiều business role thì business collaboration là kết quả của điều này. Một business collaboration là một tập những role,thực hiện một tập những ứng xử bên trong một tổ chức. Không tóm lượt một tập các role 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(interaction) giữa những role. Nó có thể được xem là một “vitual role”

Định nghĩa:

Business collaboration là hình dạng tạm thời của hai hay nhiều business role tham gia vào một tập những ứng xử trong một ngữ cảnh đặc biệt nào đó.

Ký hiệu minh họa:



Tổng quát Business Collaboration

Business collaboration có thể bao gồm một tập những business role

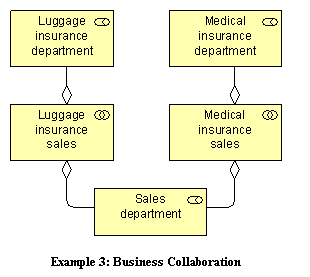
Business collaboration có thể gán với một hay nhiều business interaction

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

Hình minh họa:



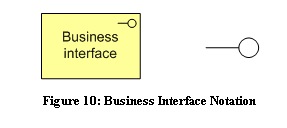
Business Interface

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

Định nghĩa:

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

Ký hiệu minh họa:



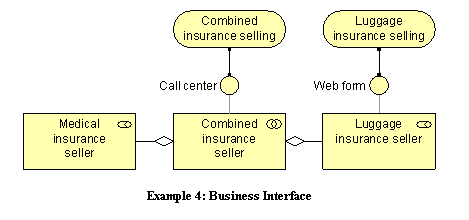
Tổng quát Business Interface:

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 thể hiện là danh từ

Hình vẽ minh họa:



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 business

Business Object

Định nghĩa:

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 business.

Tổng quát Business Object:

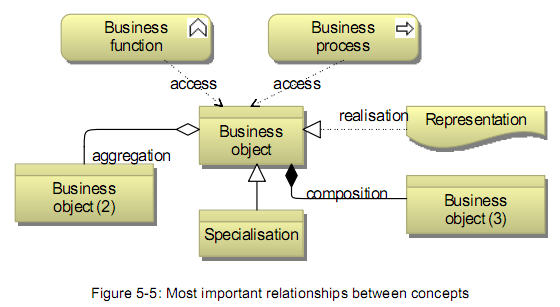
Business object có thể được truy xuất bởi business function, business process, business event, business service,

Một business object có thể hiện thực, 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

Hình vẽ minh họa:



Behavioral Concepts-Khái niệm thuộc về ứng xử

External Behavior

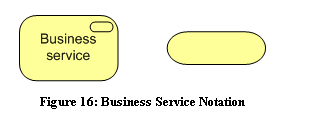
Business service

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

Định nghĩa

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

Ký hiệu:



Tổng quát 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

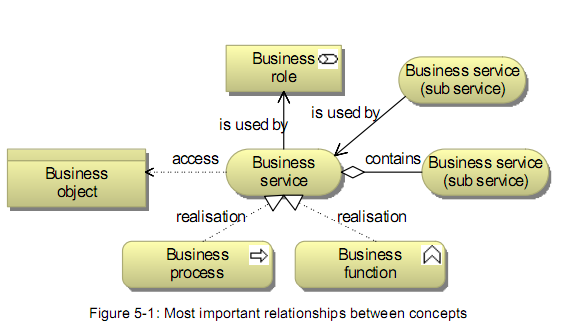
Một business service có thể bao gồm nhiều business service khác và có thể sử dụng những business service khác.

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”.

Hình minh họa:



Internal Behavior

Bên trong tổ chức, dịch vụ của tổ chức được hiện thực bởi những business behavior, 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

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

Định nghĩa

Là cấp độ nhỏ nhất của business behaviour

Ký hiệu



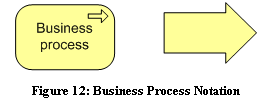
Business process

Business process miêu tả workflow hay value stream, bao gồm những process hay những 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 “customer to customer” và những customer này là những internal customer, hay nói cách khác nó là những sub-process bên trong một tổ chức. Giữa business process và business function có mối quan hện ngầm many-to-many, ví dụ process miêu tả một “flow” của những activity, fuction lại nhóm những activity lại theo một số tiêu chí như tài nguyên, kiến thức, kỹ năng …

Định nghĩa:

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 products và service

Ký hiệu



Tổng quát Business Process

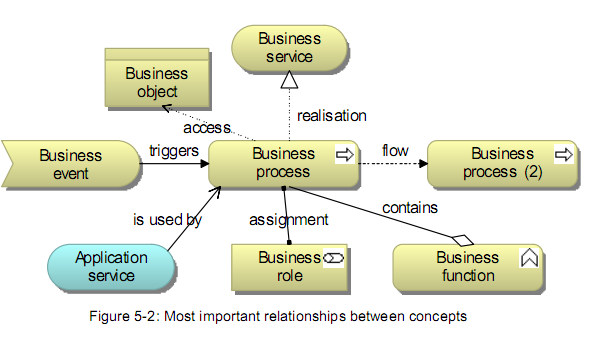
Một business process có thể kích hoạt hoặc bị kích hoạt bởi những đối tượng thuộc khía cạnh business behavior như: business event, function , interaction hay business process khác

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 đơn giản

Hình minh họa:



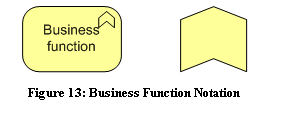
Business Function

Business function quy về chức năng có thể hữu ích cho business process, nó nhóm những ứng xử(behavior) dựa trên các tiêu chí như : năng lực, tài nguyên, kỹ năng.Trong khi business process của tổ chức được định nghĩa dựa trên product và service mà tổ chức đó quan tâm thì business function được dựa trên mối quan hệ tài nguyên đến nhiệm vụ, và appliction support

Định nghĩa

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 role bên trong tổ chức.

Ký hiệu



Tổng quát 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

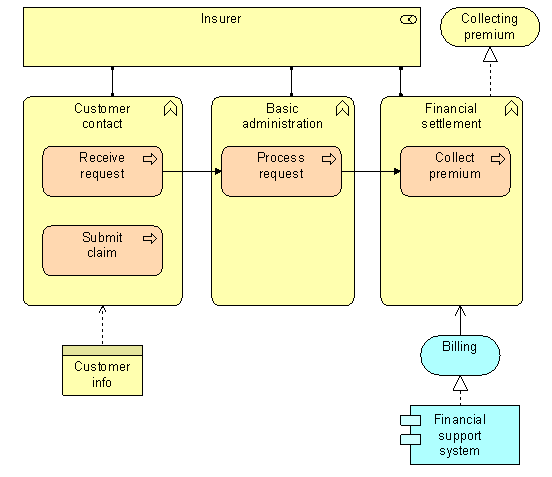
Một business function có thể truy xuất business object

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 process nên được đặt là một động từ đơn giản

Hình minh họa



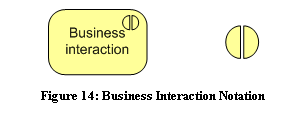
Business interaction

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

Định nghĩa:

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

Kí hiệu minh hoạ:



Tổng quát 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 iteraction có thể truy xuất business object

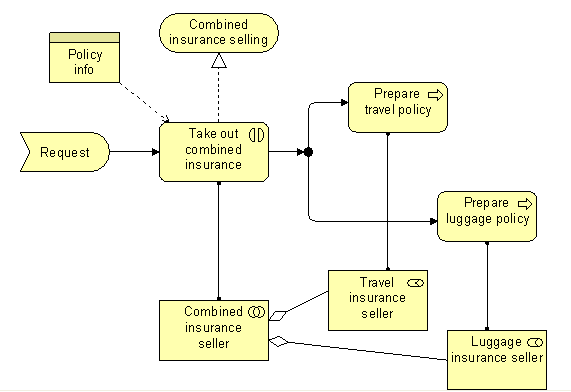
Business interaction có thể hiện thực một hay nhiều business service

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 trong thì hiện tại

Hình minh hoạ:



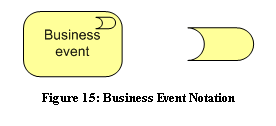
Business Event

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(model) điều gì gây(trigger) nên business function, business process, ngoài ra còn có một số kiểu khác của event có thể được dùng như: event xảy ra dẫn đến ngắt(interrup) một business process.

Định nghĩa

Business event là một điều gì đó diễn ra ở bên ngoài và tác động lên business process, business function, business interaction.

Ký hiệu minh hoạ



Tổng quát business event:

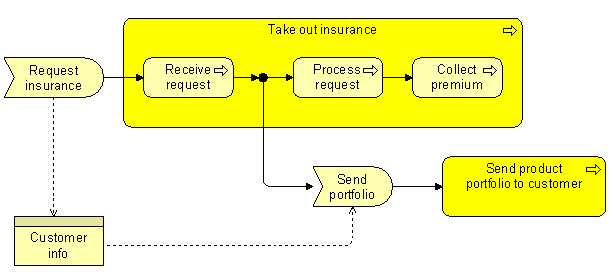
Một business event có thể gây ra hoặc được gây ra bởi business process, business function, business interaction

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

Hình vẽ minh hoạ



Informational Concepts-Khía cạnh thông tin

Trái ngược với khía cạnh cấu trúc(structure concept) và khía cạnh ứng xử (behavioral concept)- khía cạnh liên quan chủ yếu đến góc nhìn chức năng(operational perspective) trong enterprise. Khía cạnh thông tin(information concept) tập trung vào góc nhìn mà chúng ta gọi là “intentional”. 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 sản phẩm mà tổ chức đem đến cho khách hàng .

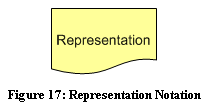
Presentation

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…)

Định nghĩa:

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

Ký hiệu:



Tổng quát về Presentation

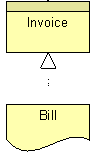
Presentation có thể hiện thực một business object

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

Hình vẽ minh hoạ



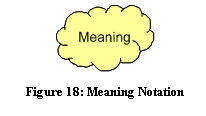
Meanning

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 user hay một loại user cụ thể nào đó.

Định nghĩa

Là sự đóng góp(của presentation) của business object vào knowledge hay expertise của một vài actor trong một hoàn cảnh đặc biệt nào đó

Ký hiệu minh hoạ

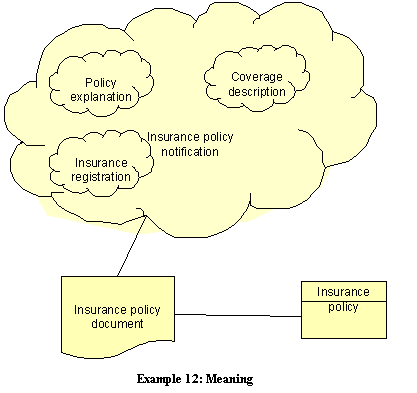


Tổng quát meanning

Một meanning có thể kết hợp với một presentation

Tên của meanning nên đươc thể hiện là danh từ hoặc cụm danh từ

Hình vẽ minh hoạ



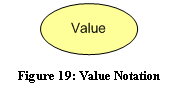
Value

Khái niệm Value của một product hay một service 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 product hay những service này. Hay nó là những gì mà những khách hàng đạt được khi nhận hay nói cách khác là “mua” những product hay service này.

Định nghĩa

Value là những thứ giúp những nhóm người liên quan đánh giá đúng một product hay một service.

Ký hiệu:



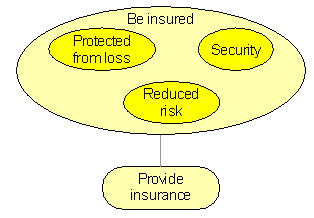
Tổng quát value

Value có thể kết hợp(association) với những service, những product(nó có thể là một phần của product), những 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 service thì nên được đặt như là một hành động-một kết quả đạt được

Hình vẽ minh hoạ



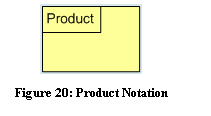
Product

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. “Buying” một product sẽ cho khách hàng quyền để sử dụng những service kết hợp với nó.

Định nghĩa

Product là một tập hợp của những service cùng với contract-bản quy định về những đặc trưng, quyền lợi, và yêu cầu sử dụng những service này

Ký hiệu



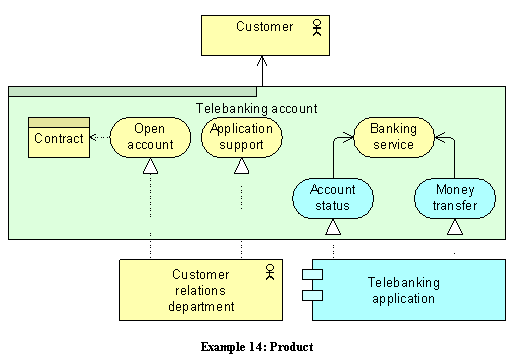
Tổng quát Product

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

Hình vẽ minh hoạ



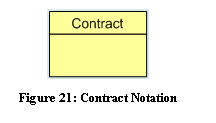
Contract

Khái niệm contract thường được sử dụng để mô hình(model) 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 service của product.

Định nghĩa

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

Ký hiệu



Tổng quát Contract

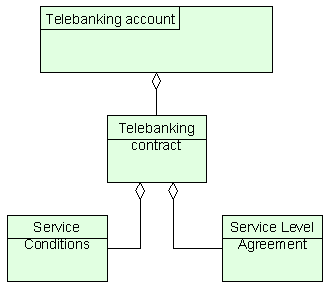
Một contract là một sự chuyên môn hoá của business object, những mối quan hệ áp dụng business object cũng có thể áp dụng cho contract

Một contract có thể chứa những contract khác

Một product có thể chứa một contract

Tên của contract nên được đặt là danh từ

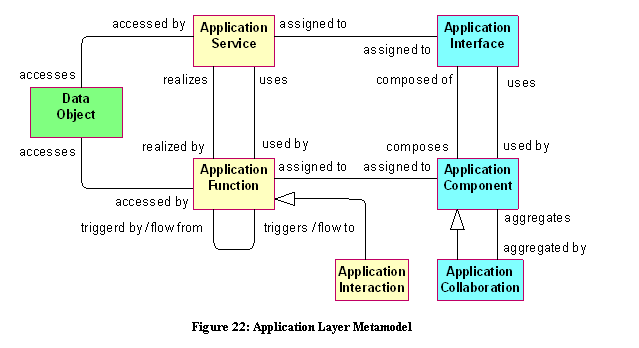
Hình vẽ minh hoạ:



* + 1. **Tầng Application**

Application Layer

Application Meta Model



Ở cấp độ high level các layer trong ngôn ngữ archimate đều có chung các khía cạnh thể hiện:

Khía cạnh structural concept:

Khía cạnh active concept

Khía cạnh passive concept hay khía cạnh data

Khía cạnh behavioral concept

Structual Concepts-Khái niệm thuộc về cấu trúc

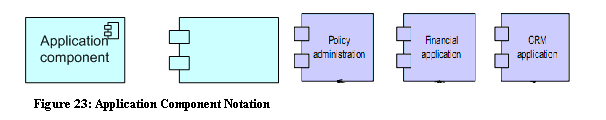
Application Component

Khái niệm chính quan trọng nhất trong khía cạnh structural concept là application component. Khái niệm được sử dụng để mô hình(modelling) và kiến trúc mọi structual entity( thực thể mang tính cấu trúc) trong application layer: Những thực thể này không chỉ là software component(thành phần phầm mềm-là một phần của application) mà nó còn có thể là một sotfware application( phần mềm ứng dụng) hoàn chỉnh, hay một sub-application, một information system( hệ thống thông tin).

Định nghĩa:

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

Kí hiệu minh hoạ:



Tổng quát application component:

Một appliction 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 (thông qua mối quan hệ aggregation)

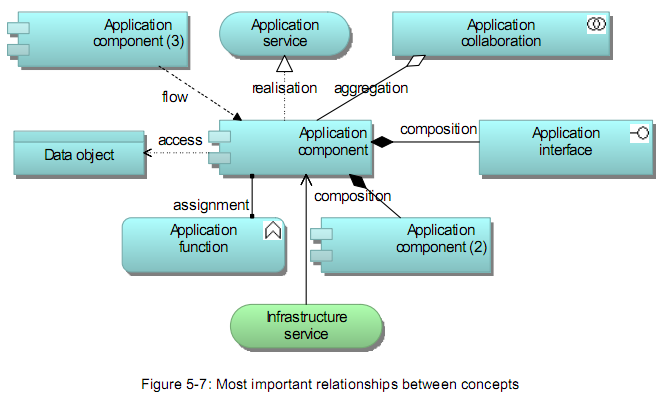
Application component có thể bao gồm nhiều application component khác(thông qua mối quan hệ composition)

Một appliction component có thể được assign với một hay nhiều appliction function, business process, business function

Một data flow có thể tồn tại giữa các application component( thông qua flow relationship)

Tên của một application component nên được khai báo là một danh từ.

Hình minh hoạ:



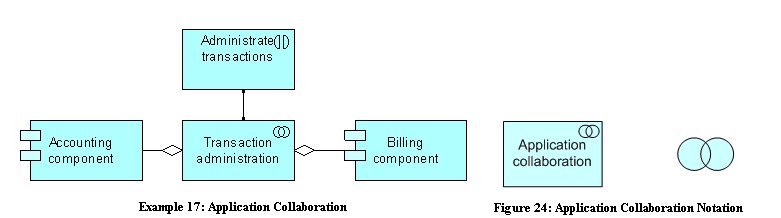
Application Collaboration

Cũng như bussiness layer, trong kiến trúc application(application architecture), mối quan hệ giữa các 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 application interaction( tương tác) ,khái niệm này cũng tương tự như collaboration được định nghĩa trong UML 2.0

Định nghĩa:

Application collaboration được định nghĩa như là một dạng hình thể tạm thời của 2 hay nhiều application component cùng tham gia vào một hay nhiều quá trình tương tác ứng dụng( application interaction).

Kí hiệu minh hoạ:



Tổng quát 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

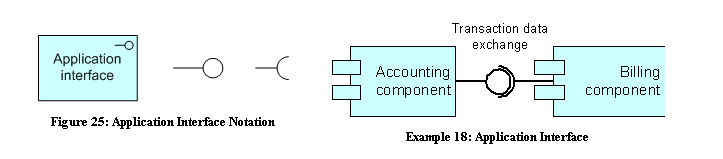
Application Interface

Application interface được xem như là một channel( kênh) mà thông qua nó, những service(dịch vụ) của một component có thể được truy xuất. Ở ý nghĩa rộng hơn, một application interface định nghĩa một vài phần tử đặc trưng thuộc về ứng xử(elementary behavioral characteristics): như một tập những chức năng hoặc sự kiện được cung cấp bởi 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 component. Có sự phân biệt giữa provided interface và required interface. Application interface cũng có thể được sử dụng để mô hình application-to-application interface( đưa ra những internal application service) và application-to business (đưa ra những external application service)

Định nghĩa:

Application interface định nghĩa một tập các chức năng(operation) và sự kiện(event) được cung cấp bởi component hay những chức năng và sự kiện được yêu cầu bởi môi trường.

Hình vẽ minh hoạ:



Tổng quát 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

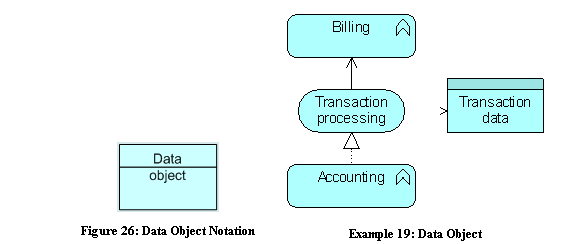
Data Object

Trong application layer, chúng ta cũng phân biệt thành phần bị động(passive counterpart) của một component. Hay còn được gọi là data object.

Định nghĩa:

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

Hình vẽ minh hoạ



Tổng quát Data Object:

Data object có thể được tạo,xoá,đọc nội dung bởi application component, application service hay application function

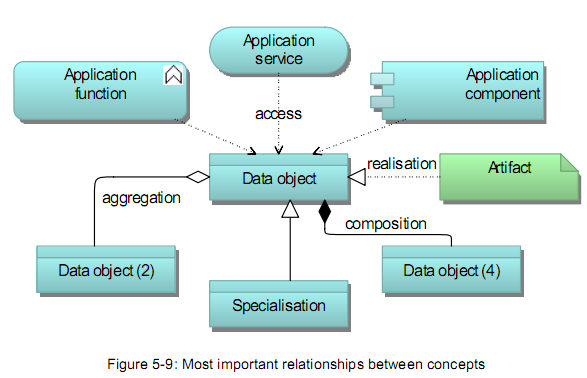
Một artifact hiện thực một data object

Một data object có thể bao gồm nhiều data object khác

Một data object có thể được specalization

Một data object nên được khai báo là một danh từ

Hình vẽ minh hoạ:



Behavioral Concepts

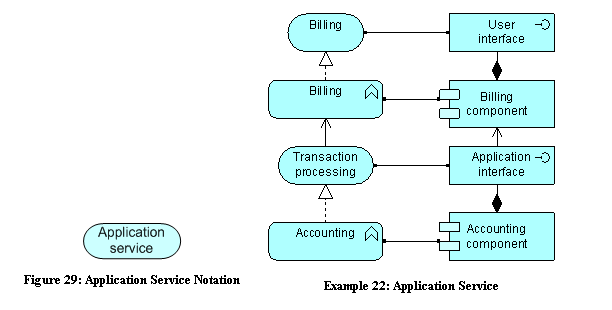
Application Service

Application service là một đơn vị chức năng có thể nhìn thấy được từ bên ngoài, nó được cung cấp bởi một hay nhiều thành phần ,được đưa ra thông qua một interface được định nghĩa rõ ràng và có ý nghĩa với môi trường(enviroment). Khái niệm service cung cấp cách mô tả rõ ràng những chức năng mà component chia sẽ với nhau và cung cấp ra môi trường bên ngoài. Interal service được đưa ra thông qua application-to-application interface. External service có thể được đưa ra thông qua application-to business interface. Application service đưa application function ra môi trường.

Định nghĩa:

Appliction 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, được đưa ra thông qua những interface được định nghĩa rõ ràng, và có ý nghĩa với môi trường.

Hình vẽ minh hoạ:



Tổng quát 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

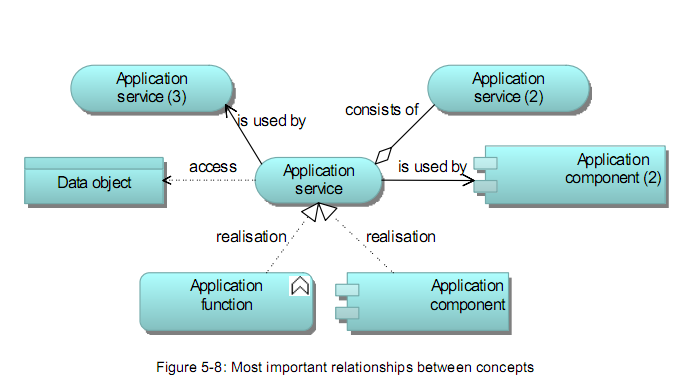
Một application service có thể được sử dụng bởi application component

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” vào cuối hoặc thêm từ “service”.

Hình minh hoạ:



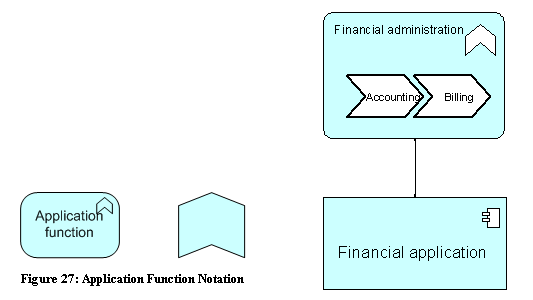
Application Function

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 services. Và lưu ý, trong architectural description 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.

Định nghĩa

Application function là ứng xử bên trong của một application component, nó hiện thực một hay nhiều application services

Hình vẽ minh hoạ:



Tổng quát 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 infrastructure layer

Tên của application function nên được khai báo là động từ thêm “-ing” ở cuối.

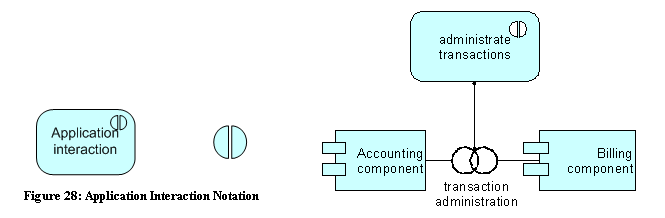
Application Interaction

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

Định nghĩa:

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

Hình vẽ minh hoạ:



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 service và của 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 Technology**

**Node, device, system software và infrastructure interface**

Structure của tầng Technology có thể được mô hình sử dụng những khái niệm node , device , system software và infrastructure interface

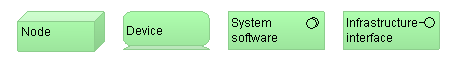
Ngôn ngữ ArchiMate đã định nghĩa **node** là một thể hiện của thực thể có cấu trúc trong tầng technology . Ví dụ như “application server” ,“database server”, “UNIX server”, “firewall”, “mail server” and “web server”. Sử dụng node , chúng ta có thể miêu tả cả tài nguyên máy tính vật lý (máy tính) , cũng như phần mềm (hệ điều hành , phần mềm ứng dụng) cho phép các chức năng .

**Device** là một chuyên môn của node , nó là một tài nguyên máy tính vật lý , ví dụ như mainframe , PCs , router , và “load banlancers” . Một device là một thực thể vật lý và không thể hiện trong bất kỳ cách nào các chức năng ngữ nghĩa (phần mềm, ứng dụng) mà thực hiện trên thiết bị.

Một chuyên môn của node là khái niệm system software , đại diện cho một môi trường phần mềm trong đó các thành phần và các đối tượng có thể được sử dụng như những artefact . Trái ngược với một device , system software là một trừu tượng của tài nguyên tính toán vật lý. Ví dụ như hệ điều hành , hệ thống database , CORBA , J2EE và phần mềm cho ứng dụng server.

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

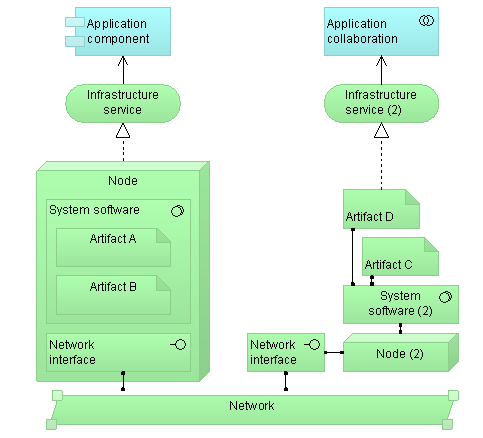
Một **infrastructureinterface** (còn được gọi là technology-interface) cho thấy cách node làm cho chức năng nó sẵn có với môi trường : những node khác , application component và application colloboration . Bạn có thể mô tả các thỏa thuận và điều kiện , cả hai bên phải tuân theo để sử dụng các chức năng lẫn nhau của họ.



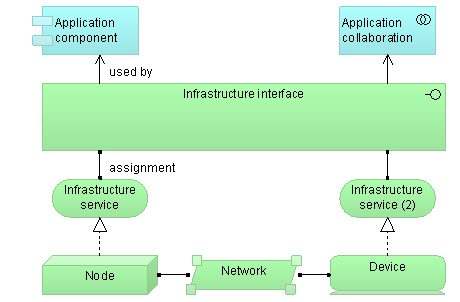
Network có thể sử dụng để kết nối những node với nhau , artefact có thể được gán cho một node , một node tồn tại nhiều node con . Devices và system software là những node , vì thể một node có thể tồn tại những device và system software khác.

* System software có thể được gán đến một device và đến một node (phụ thuộc vào bạn ưu tiên cái nào hơn) . Thay vì gán những artefact cho một node , quan hệ này có thể hiện thực cụ thể bằng cách gán một artefact cho system software .
* Bằng cách "nesting” bạn có thể trừu tượng từ các loại mối quan hệ chính xác (tập hợp, thành phần, gán).
* Một node có thể sử dụng hoặc thực hiện dịch vụ technology. Các dịch vụ technology có thể được sử dụng bởi các thành phần ứng dụng và hợp tác. Một dịch vụ technology có 'truy cập' đển artefacts
* Device có thể được kết nối với network, tương tự như node.

Hình bên dưới cho thấy sự khác biệt giữa làm nesting (trái) và assigning (bên phải). Ở bên phải, các mối quan hệ đã được làm mô hình chính thức như quan hệ gán . Ở bên trái, nó không phải là mô hình chính thức cho dù nó liên quan tập hợp, thành phần hoặc gán.



Một node có một hoặc nhiều interface. Interface có thể được sử dụng bởi các node bên ngoài tầng Technology và các thành phần và hợp tác bên ngoài lớp ứng dụng. Hơn nữa, dịch vụ Technology có thể được giao cho một interface-technology



Cách đặt tên: một tên node nên chỉ ra các loại máy tính được sử dụng; sử dụng một danh từ. Một thiết bị dùng để chỉ loại phần cứng, và hệ thống phần mềm dùng để chỉ các loại phần mềm; cho cả hai, tốt nhất sử dụng danh từ.

**Network and communication path**

Networks and communication paths cũng là những thành phần cấu trúc của tầng technology , để mô hình các khía cạnh trao đổi .

Một **communication path** là quan hệ giữa hai hoặc nhiều node, được sử dụng để trao đổi thông tin giữa các node.  
  
Một **network** là 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. Network ví dụ là: mạng không dây, Ethernet, Surfnet, mạng nội bộ, internet, truyền thông và đường dẫn đơn giản là một 'dây điện thoại. các mạng vật lý có tính chất vật lý như băng thông, trì hoãn, vv Với kiến trúc sư, những tài sản này có thể được mô hình như là thuộc tính (tài sản) nếu cấu thích hợp được xác định.

Network và communication path có thể được hình dung như hình dưới

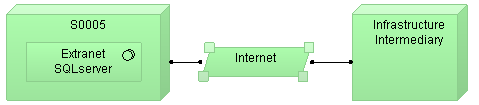


Nodes có quan hệ với nhau bằng các communication paths



Cách đặt tên: Tên interface technology phải là một danh từ.

Networks có thể được giao cho các node (và do đó với các thiết bị và phần mềm hệ thống): các node sử dụng mạng để trao đổi dữ liệu.



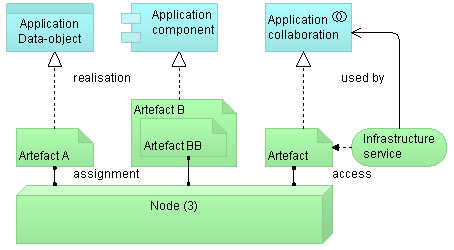
Artifact

**Artifact** là 1 khái niệm cấu trúc bị động mô hình thông tin hoặc dữ liệu , tương tự như data object và business object.   
  
Ngôn ngữ định nghĩa ArchiMate artifact như là một mảnh vật chất của thông tin được sử dụng bởi (hoặc được sản xuất bằng) một quá trình phát triển phần mềm hoặc khai thác và bảo trì hệ thống.  
  
Trong thực tế, một artifact là một khái niệm cụ thể trong thế giới vật lý và trong tầng technology là bạn đồng hành của một data object luận lý trong tầng application . Một data object là một đại diện luận lý của một khái niệm vật lý trong tầng business (business object) . Ví dụ về artifact là database tables , source files , scripts . executables . tài liệu , file mô hình và đặc tả .

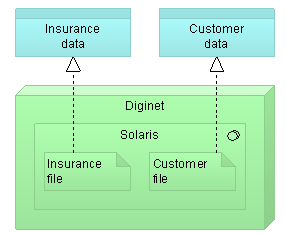
Artifact có thể được hình dung như mô tả dưới đây:



Một artifact có thể tồn tại trong nhiều artifact con , có thể nhận ra một data object , một thành phần ứngdụng và một application collaboratio . Một artifact có thể gán cho một node hoặc system software .



Cách đặt tên: artifact nên là một danh từ. Nếu vật phẩm đại diện cho một tập tin cụ thể, cơ sở dữ liệu một bảng cụ thể hoặc một tài liệu cụ thể, tốt nhất sử dụng tên giống như tên các thực thể vật lý của chúng .



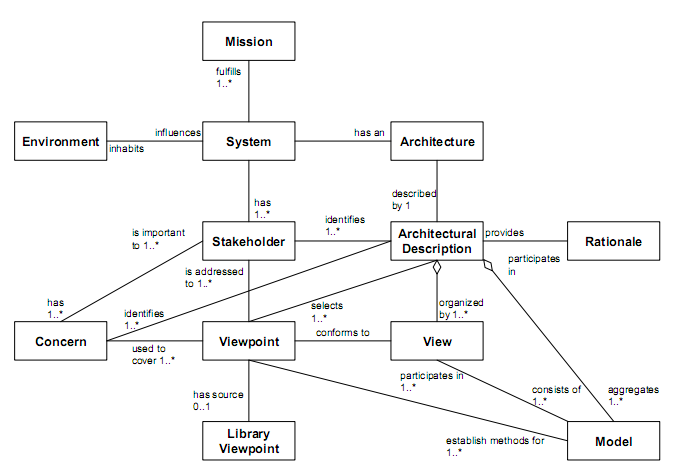
* + 1. **Viewpoint**

ViewPoint Concept

Viewpoint là gì-Tại sao lại sử cần Viewpoint

Việc xậy dựng, duy trì và phát triển một enterprise architecture 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 stackholder, 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 stackholder và những kiến trúc sư(architect) có 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 (abstraction) một tập những mô hình (models) để thể hiện một enterprise architecture, mỗi một tập mô hình trên đều hướng về một loại stackholder với những mối quan tâm(concern) cụ thể. Viewpoint có thể được sử dụng để thấy được những khía cạnh độc lập, cũng như là mối quan hệ giữa hai hay nhiều khía cạnh với nhau.



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 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.

Architecture 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 stackholder cũng được thống nhất thông qua những “presentation” phù hợp.

Khái niệm Viewpoint không chỉ giới hạn chỉ đối với cộng đồng hệ thống thông tin( information system community) mà 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 frameworks tổng quát đã được ra đời và phát triển(Finkelstein et al. 1992, Kotonya and Sommerville 1992, Nuseibeh 1994), trong frameworks 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(perspective) 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

Định nghĩa: ***Viewpoint là sự quy định rõ ràng của những thoả thuận về việc 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 stackholder với tất cả những người tham gia vào việc truyền thông(communication). Viewpoint được thiết kế cho mục đích phục vụ như là một phương tiện trong quá trình truyền thông của một cuộc hội thảo(conversation) về 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 stackholder.

Model, View, Visualization

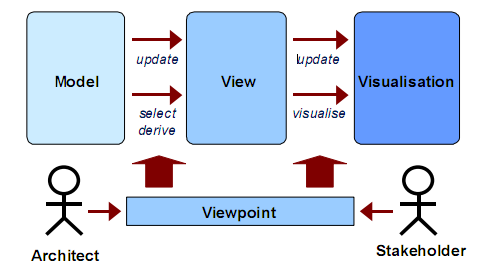
Để có thể thiết kế, chọn lựa, và sử dụng những viewpoint một cách đúng đắng, trước hết chúng ta cần phải nắm giữ được những khái niệm và những nguyên tắt quan trọng liên quan đến model, view, visualization.

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

*Content:* Thường được gọi là view, view là sự lựa chọn bắt nguồn từ symbolic model của một kiến trúc, và được thể hiện ở những dạng tương tự như những khái niệm được dùng để mô hình.

*Presentation:* 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 stackholder.

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:

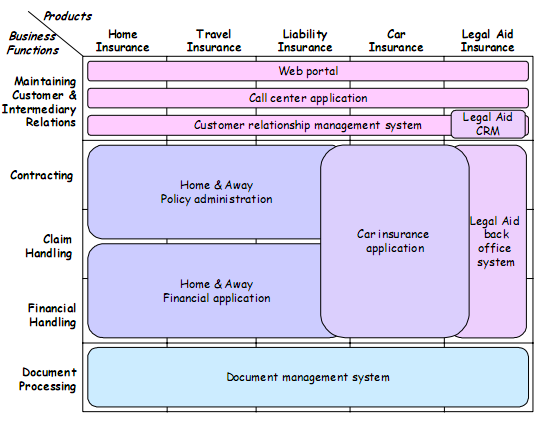


Để minh hoạ cho sự khác biệt giữa view và visualization chúng ta có thể xem hình minh hoạ bên dưới:



Với hình minh hoạ trên, ta thấy được một số những nguyên tắt presentation có thể được áp dụng vào “model-to-illustration”. Thấy được rằng những nguyên tắt này nhấn mạnh điều quan trọng trong visualization view là có thể tìm được những ký hiệu tượng trưng sinh động, phù hợp và trực quan có thể 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 enterprise architecture. Chúng thể hiện những phần tử ở dạng ‘map’ 2D dễ hiểu. Landscape map view sẽ cung cấp cho những người quản lý, những stackholder 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.



Hình trên là một ví dụ đặc trưng về landscape map, ta thấy rằng đối với một hệ thống với rất nhiều quy trình, đưa ra những sản phẩm(product) và những dịch vụ(service) đa 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 năng(business function) mà hệ thống bảo hiểm ArchiSurance cung cấp bằng việc đưa ra những product 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 stackholder và những kiến trúc sư(architect). Vì vậy việc nắm bắt những khái niệm này là một đòi hỏi cấp thiết.

Creating, Selecting, Using Viewpoint

Việc tạo, chọn lựa và sử dụng viewpoint là 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 stackholder 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 stackholder 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 dimension chính đó là *purpose* và *content*. *Purpose* là điều mà những viewpoint này phải phục vụ, và *content* là những gì mà chúng phải thể hiện ra. Dưới đây là ba kiểu kiến trúc được hỗ trợ định nghĩa theo dimension *purpose*:

Designing: Thiết kế những viewpoint hỗ trợ kiến trúc sư(architecture) và nhà thiết kế(designer) trong quá trình thiết kế từ phát thảo cho đến thiết kế chi tiết.

Deciding: Là những view hỗ trợ những nhà quản lý(manager) ra quyết định bằng cách cung cấp sự thấu hiểu sâu sắc về kiến trúc mối quan hệ giữa những domain(lĩnh vực) trong hệ thống, thường thông qua phép chiếu và phép giao của mô hình bên dưới.

Information: Những viewpoint này được sử dụng để thông tin với các stackholder để đạt được sự thấu hiểu, sự đồng tình và thuyết phục đối phương.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stackholder đặc trưng | Mục đích | Ví dụ |
| Designing | 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 đố BPMN, UML, flowchart, lược đồ ER |
| Deciding | Nhà quản lý, CEO, CIO | Hỗ trợ ra quyết định quan trọng | Landscape map, list, report |
| Informing | 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 *purpose*  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 stackholder khác.

Và đối với dimension *content,* chúng ta định nghĩa theo những những cấp độ trừu tường sau:

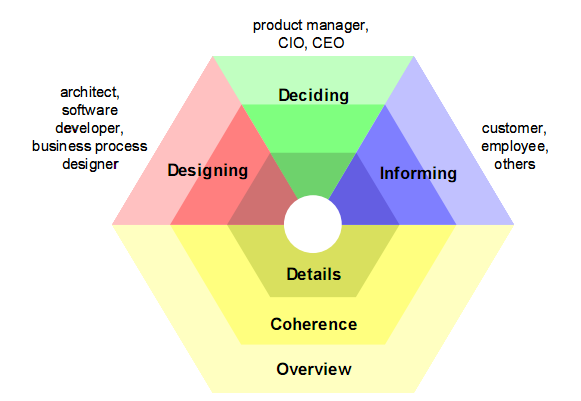
Details: Những view của cấp độ detail(chi tiết) thường là bản vẽ chi tiết về một layer, một khía cạnh(aspect) nào đó của kiến trúc, những stackholder ở đây thường là những kỹ sư phần mềm, những người trực tiếp tham gia vào thiết kế và hiện thực software component, hay có thể là người quản lý quy trình(process), là người chịu trách nhiệm về tính hiệu quả của sự thực thi quy trình .

Coherence: Những view ở cấp độ này có thể thể hiện ở nhiều layer hay nhiều khía cạnh, sự mở rộng ở nhiều layer, nhiều góc nhìn khác nhau này giúp những stackholder có thể tập trung vào những mối quan hệ trong kiến trúc, ví dụ như process-uses-system(nhiều layer), application-uses-object(nhiều khía cạnh). Những stackholder ở đây thường là nhà quản lý chức năng, những người chịu trách nhiệm về các dịch vụ IT hay quy trình kinh doanh(business process).

Overview: Tại cấp độ này, những view thường được thể hiện ở nhiều khía cạnh và nhiều layer. Những overview này thường hướng đến enterprise architect và những người ra quyết định như CEO hay CIO.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stackholder đặc trưng | Mục đích | Ví dụ |
| Details | Kỹ sư phần mềm, người quản lý quy trình kinh doanh | Thiết kế, quản lý | Lược đồ thể hiện Process, lược đồ lớp UML |
| Conherence | 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” |
| Overview | Enterprise architect, CEO, CIO | Quản lý sự thay đổi | Landscape map |

Hình bên dưới sẽ miêu tả trực quan dimension của *purpose* và cấp độ trừu tượng hoá(*content)* của viewpoint:



Sử dụng Viewpoint

Việc phân loại viewpoint dựa vào *purpose và content* 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:

Scoping: 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.

Greation of 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.

Validation: Đánh giá kết quả của việc biểu diễn những view, Xem có phải những stackholder đã đồng thuận với việc sử dụng những view này là hợp lý để biểu diễn trong ngữ cảnh này không.

Obtaining commitment: Nếu đã đạt được sự đồng thuận giữa những stackholder chính, thì bước tiếp theo là tạo những “commitment” cho kết quả đạt được.

Informing: Thông tin với những stackholder 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 lưa 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.

Basic Design Viewpoint

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:

Inwards: hướng về sự kết hợp bên trong của phần tử

Upwards: hướng về những phần tử được hỗ trợ bởi nó

Downwards: hướng về những phần tử hiện thực nó

Sidewards: Hướng về phần tử cộng tác với nó



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:



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.

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 là đây 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 | |
| Stackholder | (Enterprise) architect, nhà quản lý |
| Những Mối Quan Tâm(concert) | Lựa chọn ban đầu cho việc thiết kế hight-level overview |
| Mục Đích | Thiết kế, thông tin |
| Cấp độ trừu tượng hoá | Coherence, Overview |
| Layer | Business, application, Technology |
| Khía cạnh | Active, passive, behaviour |

Những khái niệm chính được sử dụng trong Introductory Viewpoint sẽ được thể hiện trong hình sau:



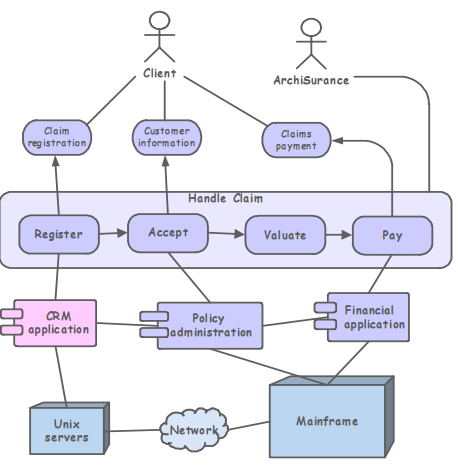
Khi sử dụng Introductory Viewpoint chúng ta sử dụng những ký hiệu đã được đơn giản và rút gọn đối với những khái niệm, cũng như là những mối quan hệ:

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.

Mục đích của chúng ta là để trách cảm giác nhầm lẫn là kiến trúc đã được thiết kế hoàn tất như đã nêu ở trên. Dưới đây là một ví dụ về Introductory Viewpoint:



Organization Viewpoint

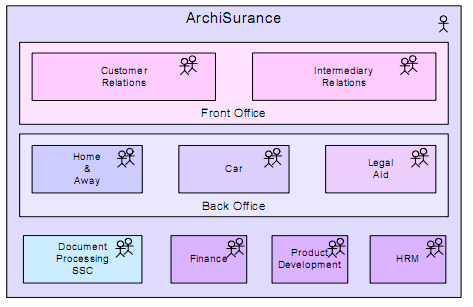
Organization Viewpoint thể hiện kết cấu bên trong một tổ chức của một tập đoàn, hay một phòng ban, một đơn vị thuộc tổ chức, điều đó tuỳ vào việc kiến trúc sư đang đứng ở vị trí nào. Chúng thường được biểu diễn bằng lược đồ gồm những khối lồng vào nhau(như hình ), ngoài ra, chúng còn có thể được biểu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

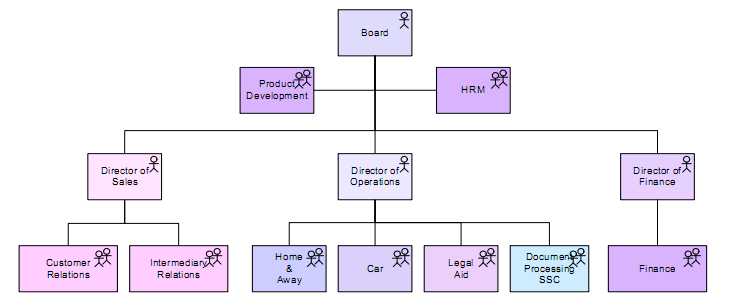
|  |  |
| --- | --- |
| Organization Viewpoint | |
| Stackholder | Enterprise architect, nhà quản lý, nhân viên |
| Những Mối Quan Tâm(concert) | Kết cấu bên trong tổ chức, về trách nhiệm và quyền lợi |
| Mục Đích | Thiết kế, thông tin, ra quyết định |
| Cấp độ trừu tượng hoá | Details |
| Layer | Business |
| Khía cạnh | Active |

Một Organization Viewpoint thường gồm những khái niệm:



Hình sau là ví dụ điển hình 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 “context diagram”, 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.

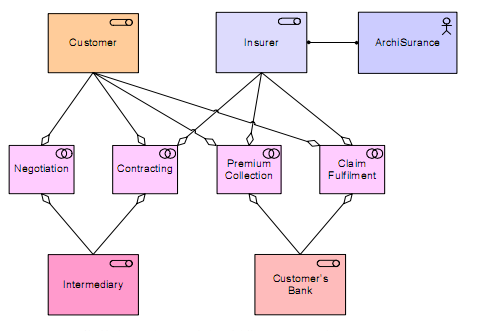
Một chứa năng quan trọng khác của viewpoint này là thể hiện làm thế nào những actor trong tổ chức cộng tác với nhau để hiện thực những business process, bằng cách thể hiện dòng chảy thông tin giữa chúng.

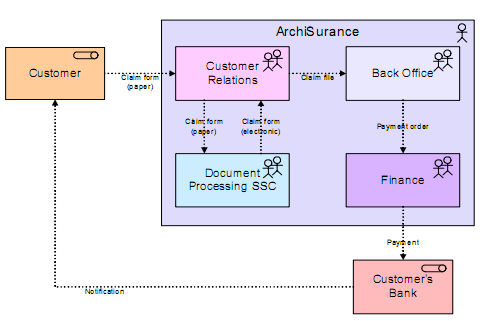
|  |  |
| --- | --- |
| Actor Cooperation Viewpoint | |
| Stackholder | Enterprise architect |
| Những Mối Quan Tâm(concert) | Mối quan hệ, và sự cộng tác của những actor với môi trường của chúng |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business |
| Khía cạnh | Structure, Behavior |

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



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





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.

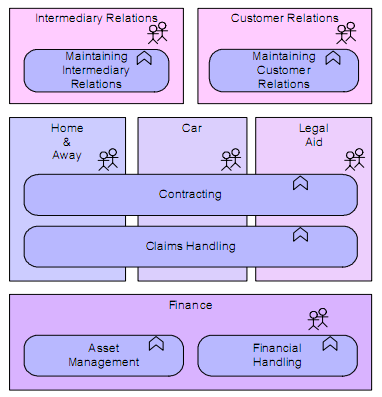
Business Function Viewpoint thường được sử dụng để cung cấp một cái nhìn ở mức high-level về những chức năng tổng quát của một công ty và để cấu trúc những hoạt động chính liên quan đến tổ chức.

|  |  |
| --- | --- |
| Business Function Viewpoint | |
| Stackholder | Enterprise architect |
| Những Mối Quan Tâm(concert) | Nhận diện những hoạt động thuộc về bản chất, giảm sự phức tạp. |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business |
| Khía cạnh | Behavior |

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



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



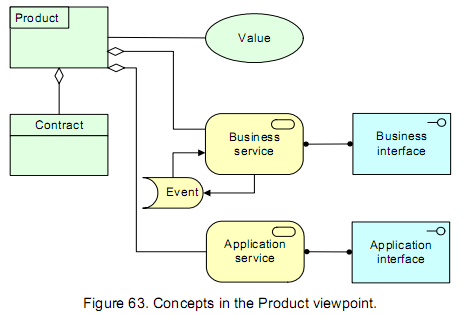
Product Viewpoint

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 service, và một contract.

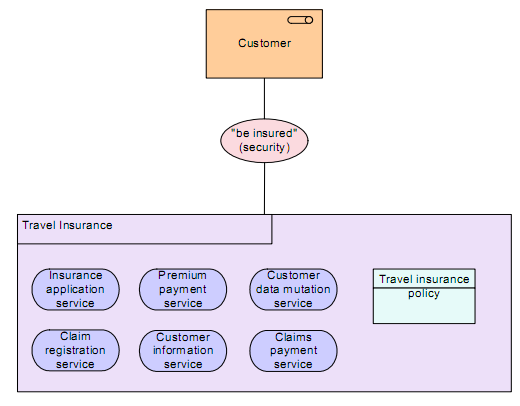
Product Viewpoint thường được trong quá trình phát triển product, chúng dùng để thiết kế product bằng cách kết hợp những những dịch vụ sẵn có, hay nhận diện những dịch vụ mới cần được tạo cho product này, với những giá trị mà khách hàng mong muốn nhận được từ nó. Bên cạnh đó, Product Viewpoint còn có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

|  |  |
| --- | --- |
| Product Viewpoint | |
| Stackholder | Người phát triển product  Người quản lý product  Kiến trúc sư |
| Những Mối Quan Tâm(concert) | Quá trình phát triển product  Giá trị được đưa ra bởi những product của một enterprise |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business ,(application) |
| Khía cạnh | Behavior, information |

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



Hình là một ví dụ về việc sử dụng Product Viewpoint



Service Realisation Viewpoint

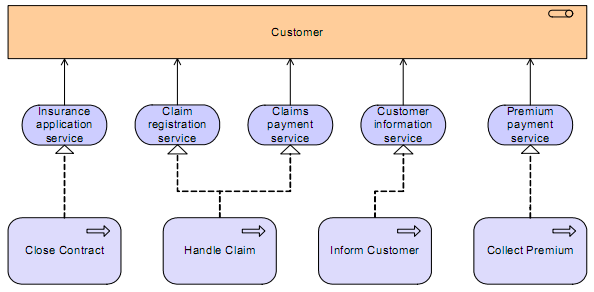
Service Realisation Viewpoint được sử dụng để thể hiện một hoặc nhiều business service được hiện thực bởi những process bên dưới như thế nào. Vì vậy, nó thường là cầu nối giữa Product Viewpoint và Business process Viewpoint. Nó cung cấp cái nhìn từ bên ngoài vào một hay nhiều business process.

|  |  |
| --- | --- |
| Service Realisation Viewpoint | |
| Stackholder | Kiến trúc sư của quy trình, nhà quản lý sản phẩm và chức năng |
| Những Mối Quan Tâm(concert) | Giá trị thêm vào của business process về tính nhất quán, tính toàn vẹn |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business ,(application) |
| Khía cạnh | Behavior |

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



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



Business Process Collaboration Viewpoint

Một Business Process Collarboration 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 process này bằng cách cung cấp sự thấu hiểu về sự phụ thuộc bên trong của những process 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 service bởi business process

Việc chia sẻ dữ liệu

Sự thực thi một business process bởi những role hay những actor giống nhau.

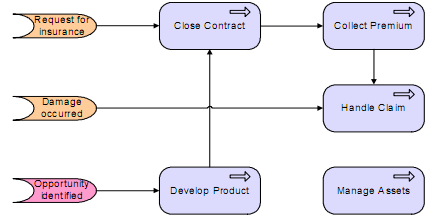
Mỗi một khía cạnh này có thể được thể hiện như là một sub-viewpoint của Business Process Collarboration Viewpoint.

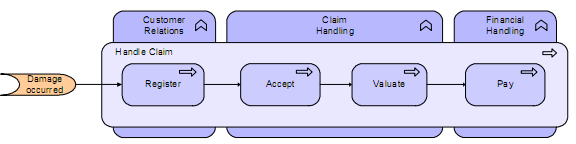
|  |  |
| --- | --- |
| Business Process Collarboration Viewpoint | |
| Stackholder | Kiến trúc sư của quy trình, nhà quản lý về chức năng |
| Những Mối Quan Tâm(concert) | Sự phụ thuộc của những business process, tính toàn vẹn, tính nhất quán, trách nhiệm |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business ,(application) |
| Khía cạnh | Behavior |

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



Hình là ví dụ về việc sử dụng 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 process có thể góp phần hiện thực một product

Sự gán kết giữa role và business process.

Thông tin được sử dụng bởi business process.

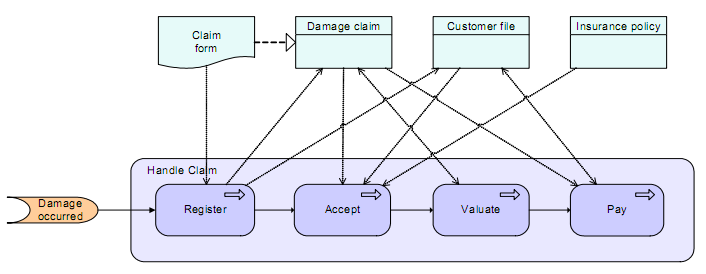
Mỗi một vấn đề trên cũng đều có thể được thể hiện như là một sub-viewpoint của một Business Process Viewpoint.

|  |  |
| --- | --- |
| Business Process Viewpoint | |
| Stackholder | Kiến trúc sư của quy trình, nhà quản lý về 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 | Details |
| Layer | Business ,(application) |
| Khía cạnh | Behavior |

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



Hình là ví dụ về việc sử dụng Business Process Viewpoint



Information Structure Viewpoint

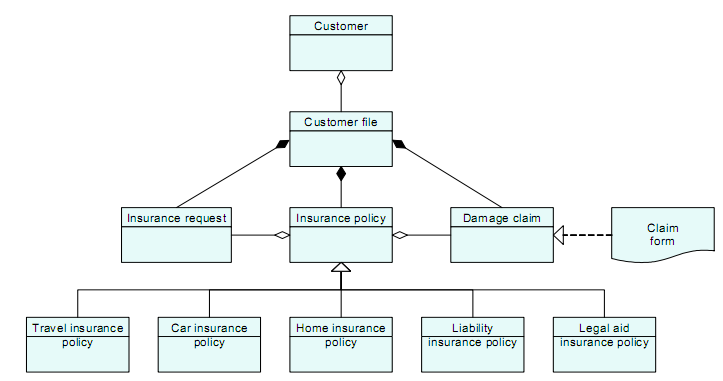
Information Structure Viewpoint là tư tưởng đúng đắn nhất để thể hiện một mô hình thông tin(traditional information model) truyền thống, được tạo trong quá trình phát triển của hầu hết mọi hệ thống thông tin. Nó thể hiệu cấu trúc thông tin được sử dụng trong một enterprise hay được sử dụng trong một 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 | |
| Stackholder | Kiến trúc sư |
| Những Mối Quan Tâm(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 | Details |
| Layer | Business ,(application) |
| Khía cạnh | Passive |

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



Hình là ví dụ về việc sử dụng Information Structure Viewpoint



Application Cooperation Viewpoint

Application Cooperation Viewpoint thể hiện mối quan hệ giữa những application hay giữa những component. Viewpoint này mô tả sự phụ thuộc trong dòng trao đổi thông tin giữa những component, hoặc mô tả dịch vụ được đưa ra hay được sử dụng bởi chúng. Viewpoint này thường được sử dụng để tạo một overview về application landscape map của một tổ chức.

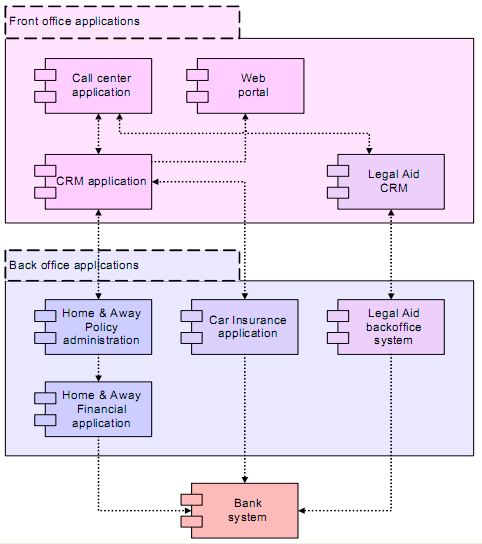
Viewpoint này cũng thường được sử dụng để thể hiện sự điều phối những application service trong việc hỗ trợ hiện thực những business process. Bằng việc mô hình những sự phụ thuộc bên trong của những service, Sự cộng tác của những application bên dưới được thiết lập trong nhiều cách độc lập hơn.

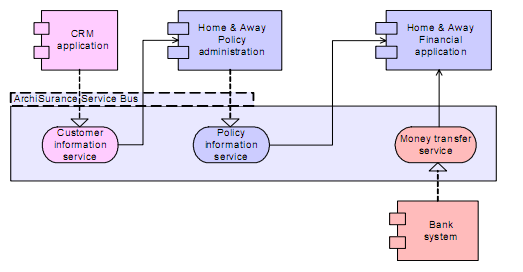
|  |  |
| --- | --- |
| Application Cooperation Viewpoint | |
| Stackholder | Enterprise architectect, architect(kiến trúc sư) |
| Những Mối Quan Tâm(concert) | Mối quan hệ và sự phụ thuộc của những application, sự điều phối service, giảm sự phức tạp. |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Coherence,Details |
| Layer | Applicaton |
| Khía cạnh | Active, Behaviour,Passive |

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



Hình là ví dụ về việc sử dụng Application Cooperation Viewpoint





Application Usage Viewpoint

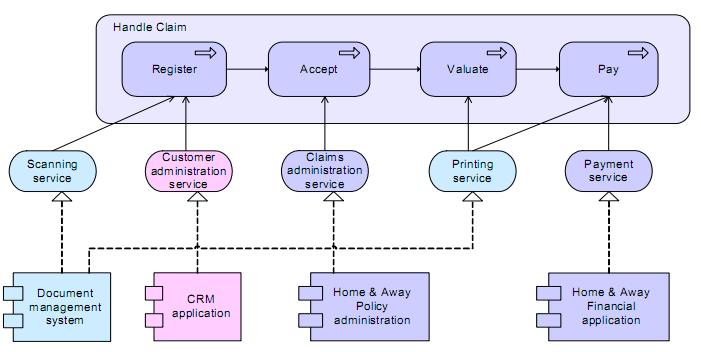
Application usage Viewpoint mô tả cách application được sử dụng để hỗ trợ một hoặc nhiều business process, và chúng được sử dụng bởi những application khác như thế nào. Viewpoint này có thể được dùng để thiết kế một application bằng cách nhận diện những service cần sử dụng bởi business process và những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

|  |  |
| --- | --- |
| Application usage Viewpoint | |
| Stackholder | Enterprise architectect, architect(kiến trúc sư) , nhà quản lý chức năng. |
| Những Mối Quan Tâm(concert) | Tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế, ra quyết định |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Business,Applicaton |
| Khía cạnh | Active, Behaviour,Passive |

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



Hình là ví dụ về việc sử dụng Application Cooperation Viewpoint



Application Behaviour Viewpoint

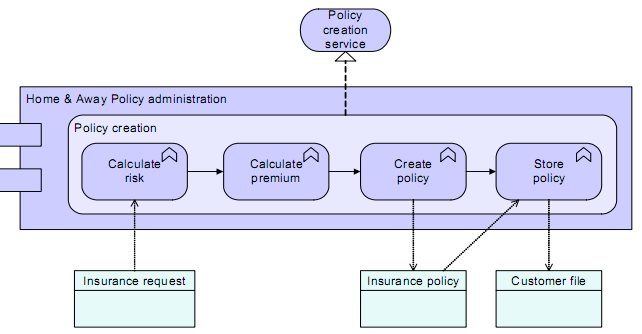
Application Behaviour Viewpoint mô tả ứng xử bên trong của application hay của những component, ví dụ, khi nó hiện thực một hay nhiều application service. Viewpoint này thường hữu ích khi khi thiết kế những ứng xử chính của application hay component, hoặc trong việc nhận diện những chức năng chồng lênh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

|  |  |
| --- | --- |
| Application Behaviour Viewpoint | |
| Stackholder | Enterprise architectect, architect(kiến trúc sư) , kỹ sư phần mềm |
| Những Mối Quan Tâm(concert) | Cấu trúc,mối quan hệ và sự phụ thuộc của những application, tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Details |
| Layer | Applicaton |
| Khía cạnh | Active, Behaviour,Passive |

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



Hình là ví dụ về việc sử dụng Application Cooperation Viewpoint



Application Structure Viewpoint

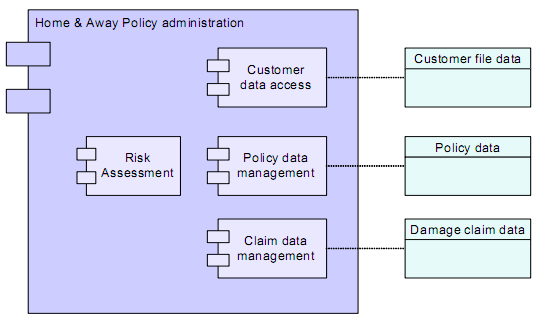
Application Structure Viewpoint thể hiện cấu trúc của một hoặc nhiều application hay component. Viewpoint này được sử dụng để thiết kế hoặc để có thể hiểu được cấu trúc chính của application hay component và dữ liệu kết hợp với chúng. Ví dụ giúp cho bước đầu trong việc phân nhỏ cấu trúc thành từng phần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

|  |  |
| --- | --- |
| Application Structure Viewpoint | |
| Stackholder | Enterprise architectect, architect(kiến trúc sư) , kỹ sư phần mềm |
| Những Mối Quan Tâm(concert) | Cấu trúc của application, tính nhất quán, tính toàn vẹn, giảm sự phức tạp |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Details |
| Layer | Applicaton |
| Khía cạnh | Active, Passive |

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



Hình là ví dụ về việc sử dụng Application Structure Viewpoint



Infrastructure Viewpoint

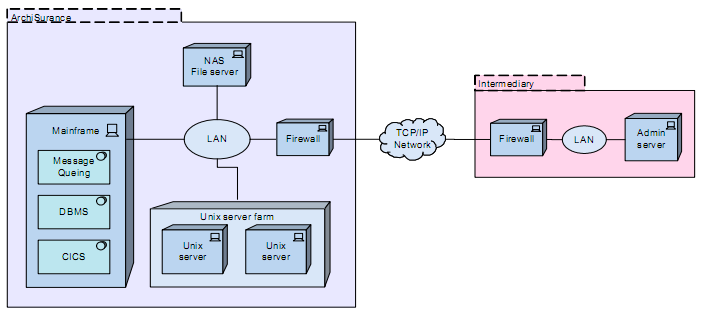
Infrastructure Viewpoint bao gồm cơ sở hạ tầng về phần cứng và phần mềm đi theo, những yếu tố mà application layer phụ thuộc vào. Nó chứa các thiết bị vật lý, hệ thống mạng, và phần mềm hỗ trợ như hệ điều hành ,cơ sở dữ liệu, middleware.

|  |  |
| --- | --- |
| Infrastructure Viewpoint | |
| Stackholder | Infrastructure architects, (application architects)  Operational managers |
| Những Mối Quan Tâm(concert) | 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 | Details |
| Layer | Technology |
| Khía cạnh | Behaviour, structure |

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



Hình là ví dụ về việc sử dụng Infrastructure Viewpoint



Infrastructure Usage Viewpoint

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

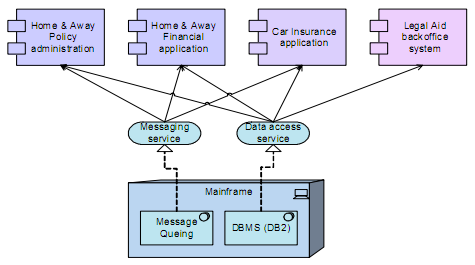
Viewpoint này đóng vai trò quan trọng trong việc phân tính hiệu quả và tính linh động của hệ thống, vì nó thể hiện sự liên quan giữa cơ sở hạ tầng vật lý với ứng dụng luận lý. Nó hữu ích trong việc xác định yêu cầu về tính hiệu quả và chất lượng của cơ sở hạ tầng dựa trên những đòi hỏi đa dạng của những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

|  |  |
| --- | --- |
| Infrastructure Usage viewpoint | |
| Stackholder | Application, infrastructure architects  Operational managers |
| Những Mối Quan Tâm(concert) | 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 | Coherence |
| Layer | Application,Technology |
| Khía cạnh | Behaviour, structure(active) |

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



Hình là ví dụ về việc sử dụng Infrastructure Usage viewpoint



Implement & Deloyment Viewpoint

Implement & Deloyment Viewpoint thể hiện cách để một application được triển khai trên cơ sở hạ tầng. Bao gồm việc ánh xạ application hay component 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 vì 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 | |
| Stackholder | Application, infrastructure architects  Operational managers |
| Những Mối Quan Tâm(concert) | Bảo mật, rủi ro, và các thành phần phụ thuộc |
| Mục Đích | Thiết kế |
| Cấp độ trừu tượng hoá | Coherence |
| Layer | Application,Technology |
| Khía cạnh |  |

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



Hình là ví dụ về việc sử dụng Infrastructure Usage viewpoint



Layer Viewpoint

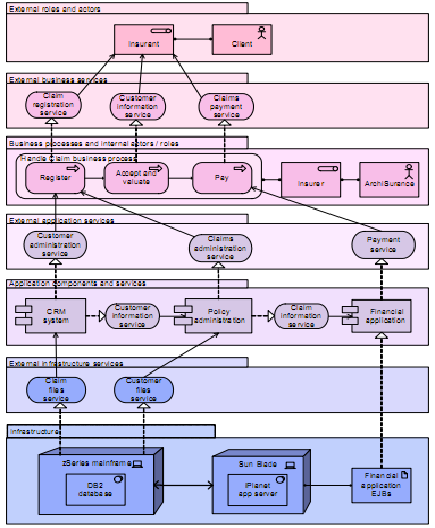
Layer Viewpoint thể hiện nhiều layer nhiều khía cạnh trong một diagram, Mục đích chính của viewpoint này là cung cấp một cái nhìn tổng quát về một phần của một business architecture trong một bức tranh. Xa hơn nữa, nó có thể được sử dụng để đánh giá tác động của sự thay đổi hoặc phân tích hiệu quả khi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

|  |  |
| --- | --- |
| Layer Viewpoint | |
| Stackholder | Application, infrastructure architects  Operational managers |
| Những Mối Quan Tâm(concert) | 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 | Overview |
| Layer | Business,Application,Technology |
| Khía cạnh | Structure, behaviour,information |

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



Hình là ví dụ về việc sử dụng Layer Viewpoint



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 stackholder khác nhau của một kiến trúc: end-user, developer, systems engineers, project managers,...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 |
| Stackholder | 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 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 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  ... |  |

* + 1. **So sánh Archimate và UML**

ArchiMate được tạo ra để mô hình kiến trúc của doanh nghiệp (tất cả các hệ thống trong một tổ chức). ArchiMate các mô hình Business, hệ thống thông tin (ứng dụng và dữ liệu), và kiến trúc công nghệ môi trường, cách kiến trúc này là liên quan đến nhau.

UML vẫn còn chức năng tốt nhất như một cách để kiến trúc của một hệ thống duy nhất .UML cung cấp 13 loại biểu đồ, cung cấp sự linh hoạt để mô tả nhiều loại khác nhau của hệ thống.

Archimate bắt đầu với một sự hiểu biết rằng những vấn đề liên quan đến nhau; rằng toàn bộ phức tạp và Business khó khăn của sự hiểu biết CNTT đòi hỏi một phong phú mối quan hệ liên lĩnh vực hoàn toàn khác nhau, từ động lực Business cho quá trình Business với các dịch vụ quản lý để hệ thống cơ sở hạ tầng. Vì vậy Archimate đi nơi UML không đi: nó định nghĩa một metamodel cho phép các mối quan hệ được xây dựng, và hạn chế, và truyền đạt. Các khó khăn cho phép phân tích, truy xuất nguồn gốc, quản trị, và nhất quán. UML là không bị giới hạn giữa các loại mô hình. Archimate thì không.

* 1. **SOA**
     1. **Giới thiệu**
     2. **Chi tiết**
  2. **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*

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.

*Yêu cầu chức năng*

Thêm mới học sinh

Cập nhật thông tin học sinh

Xem thông tin chi tiết học sinh

Xóa thông tin 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*

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.

*Yêu cầu chức năng*

Thêm mới giáo viên

Cập nhật thông tin giáo viên

Xem thông tin chi tiết giáo viên

Tìm kiếm giáo viên theo những tiêu chí cho trước

Phân công giảng dạy theo các lớp

**Quản lý việc thu học phí**

*Mô tả*

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.

*Yêu cầu chức năng*

Thêm mới học sinh nộp học phí

Cập nhật tiền học phí của học sinh

Xem chi tiết thông tin học phí của từng học sinh

Liệt kê,tìm kiếm danh sách sinh viên còn nợ học phí,đã hoàn thành học phí

**Gửi mail và tin nhắn SMS**

*Mô tả*

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.

*Yêu cầu chức năng*

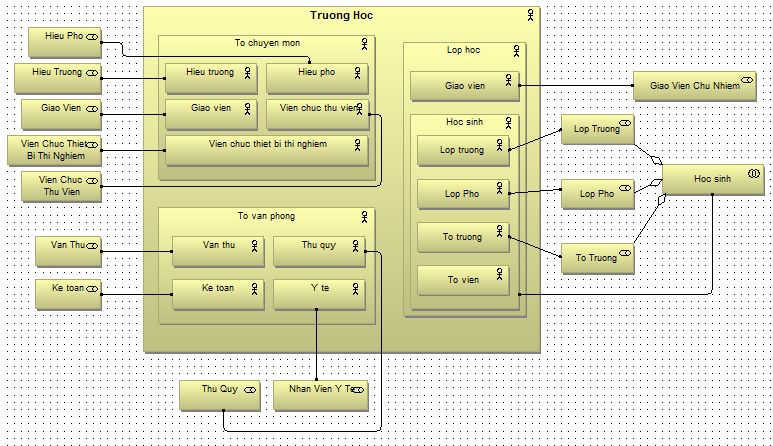
Cho phép gửi và nhận mail

Cho phép gửi và nhận tin nhắn SMS

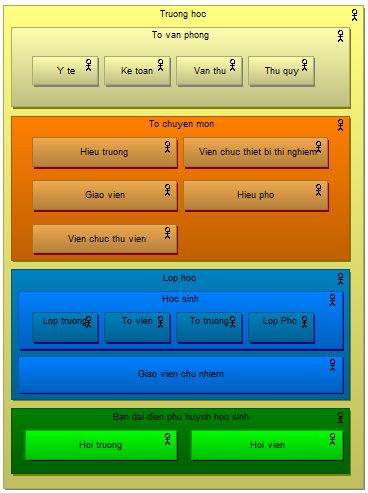
* + 1. **Mô hình kiến trúc thể hiện qua archimate**

***Bussiness layer***

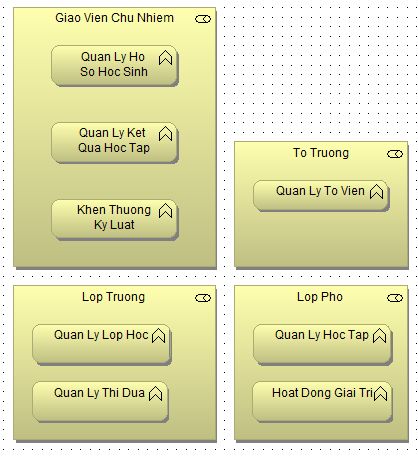
1. Bussiness Actor cooperation view

****

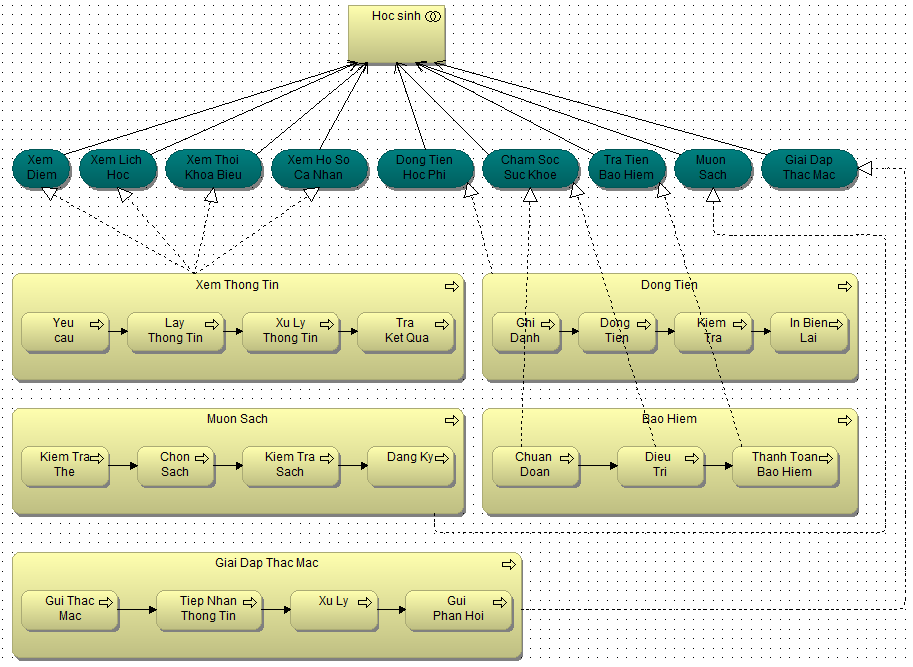
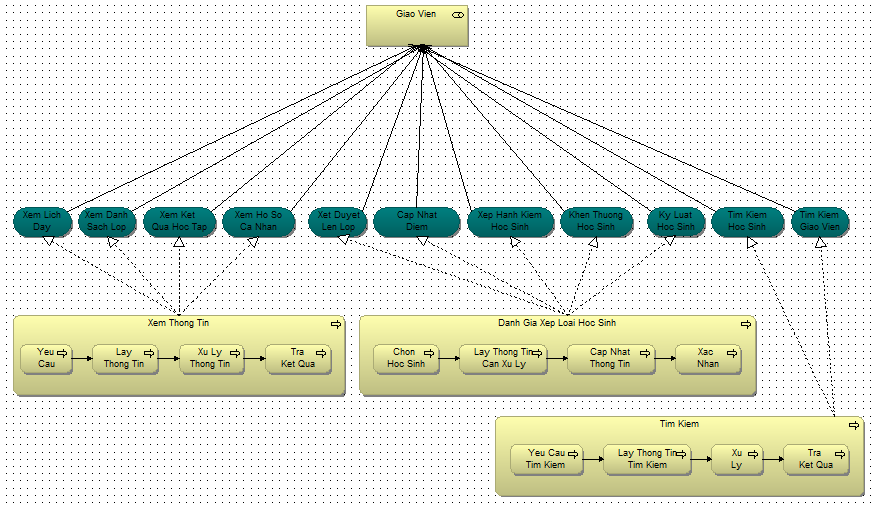
1. Bussiness Organisation structure view

****

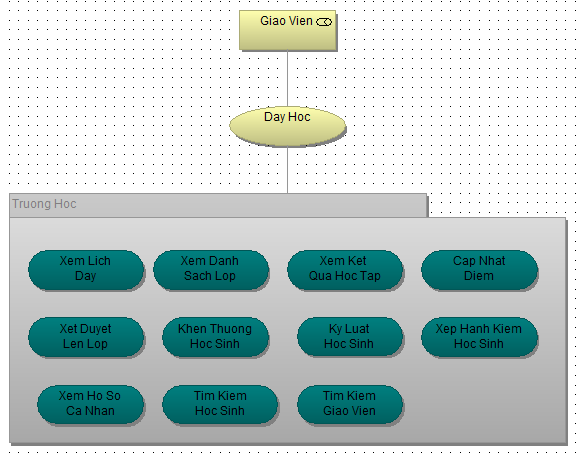
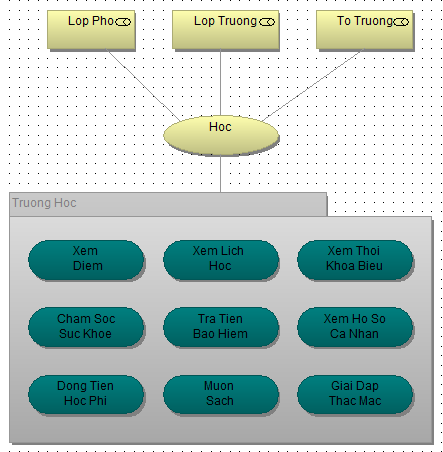
1. Business function view



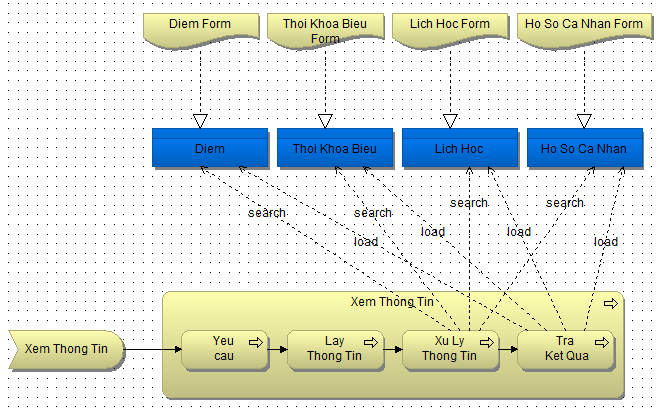
1. Business process cooperation view

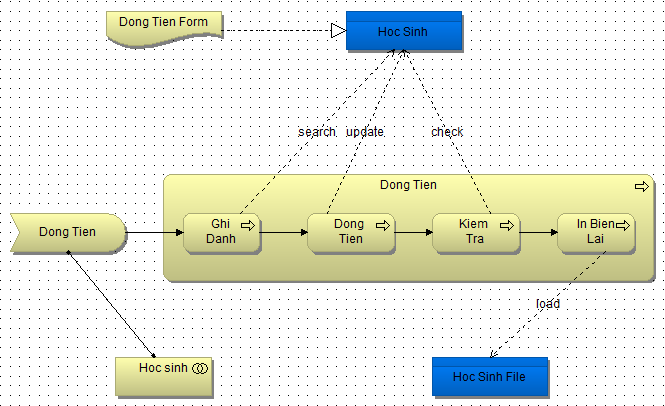


Business product view

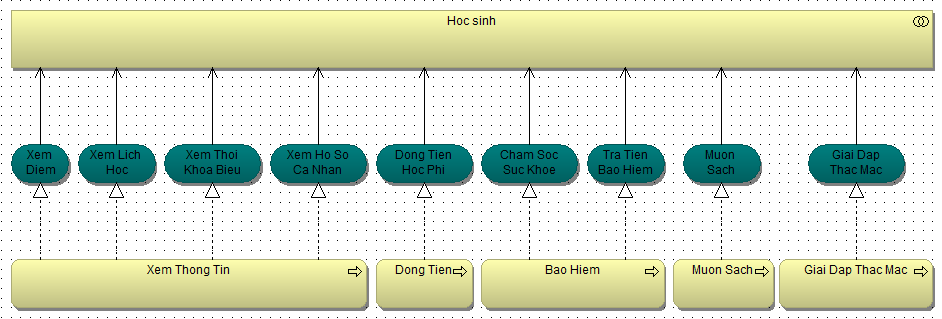
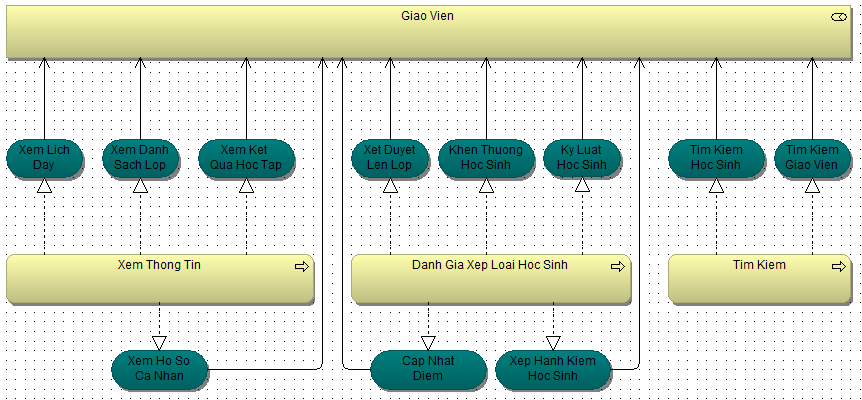


Bussiness process view



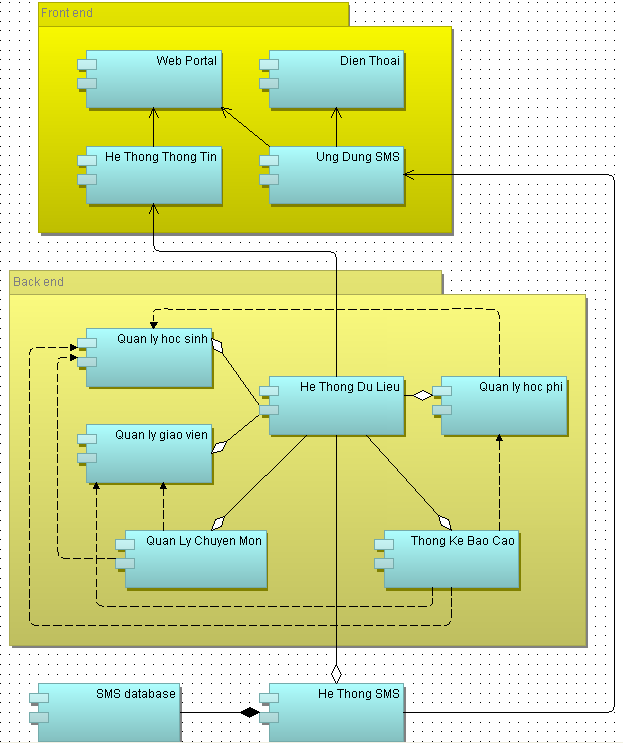


Bussiness Service realisation view

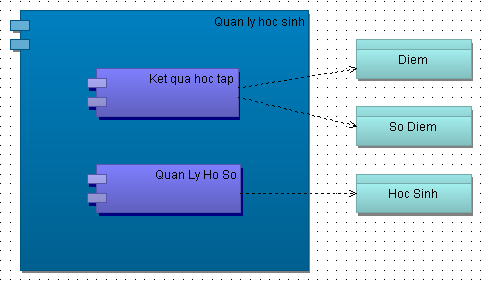
 

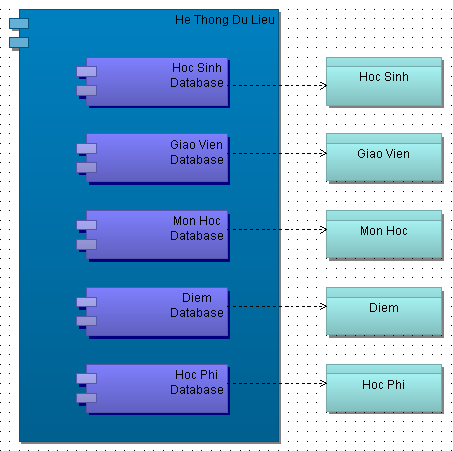
***Application layer***

Application cooperation view

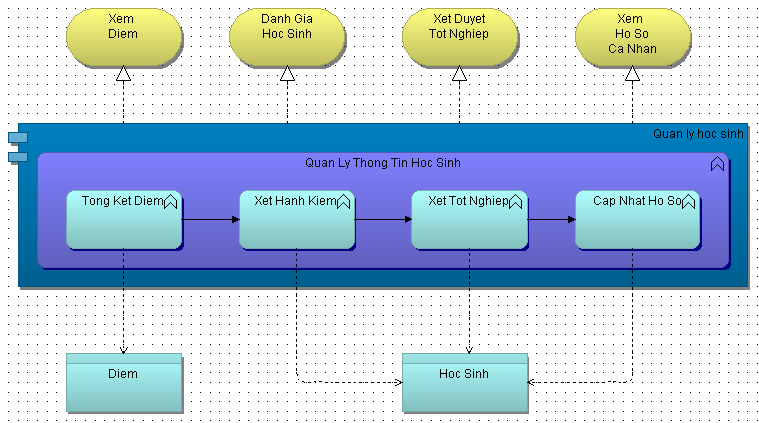


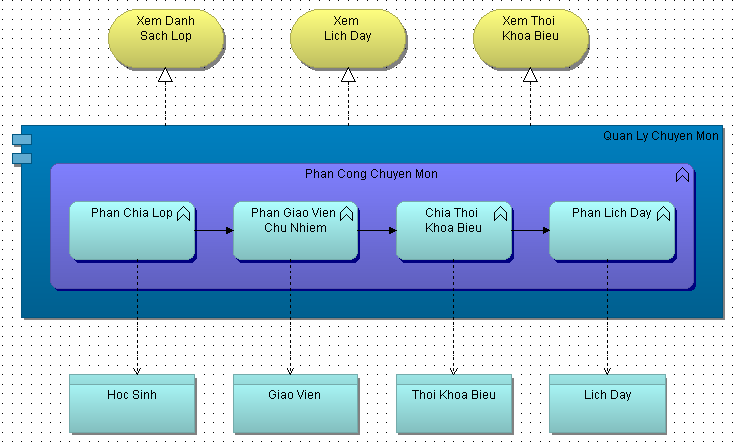
Application behavior view

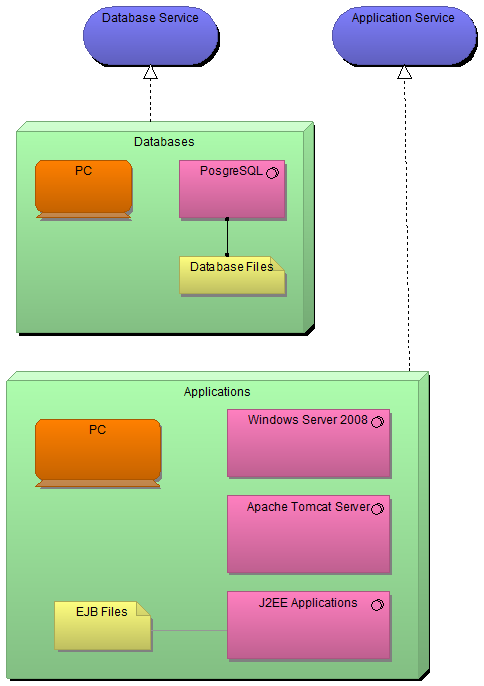




Application structure view







1. Infrastructure Viewpoint
   * 1. **Mô hình UML thể hiện chi tiết thiết kế**
     2. **Thiết kế cơ sở dữ liệu**
     3. **Chức năng quản lý giáo viên**
     4. **Chức năng quản lý học sinh**
     5. **Chức năng quản lý tiền thu học phí**

# KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN

* 1. **KẾT QUẢ ĐẠT ĐƯỢC**

…………………….

* 1. **HƯỚNG PHÁT TRIỂN**

………………………….

# TÀI LIỆU THAM KHẢO

**[1]** **Dave Chappell**, Enterprise Service Bus, thứ nhất, O'Reilly,2004,0-596-00675-6

**……………………………….**

# PHỤ LỤC