Phân Tích và Thiết Kế Hệ Thống

(IT3120)

Nguyễn Nhật Quang

quang.nguyennhat@hust.edu.vn

Trường Đại học Bách Khoa Hà Nội

Viện Công nghệ thông tin và truyền thông

Năm học 2020-2021

Nội dung học phần:

- Giới thiệu về Phân tích và thiết kế hệ thống thông tin hướng đối tượng
- Giới thiệu về Ngôn ngữ mô hình hóa UML
- Giới thiệu về Quy trình phát triển phần mềm
- Phân tích môi trường và nhu cầu
- Phân tích chức năng
- Phân tích cấu trúc
- Phân tích hành vi
- **Thiết kế kiến trúc tổng thể của hệ thống**
- **Thiết kế giao diện sử dụng**
- Thiết kế chi tiết lớp
- Thiết kế dữ liệu

Mục đích của thiết kế hệ thống

- **Phân tích** là để trả lời câu hỏi "là gì/làm cái gì" ("what") – tập trung vào các **yêu cầu (chức năng và phi chức năng)** đối với hệ thống
  - Gồm 6 bước đầu tiên (trong 10 bước) của quy trình RUP
- **Thiết kế** là để trả lời câu hỏi "làm thế nào" ("how") – tập trung nghiên cứu **sự thực thi** của hệ thống
  - Đưa ra những quyết định thiết kế phù hợp với các công nghệ được lựa chọn
  - Đáp ứng các yêu cầu phi chức năng (vd: giao diện, hiệu năng, tính sẵn sàng, tính bảo mật,…)

Thiết kế kiến trúc tổng thể

- Mục đích của thiết kế kiến trúc tổng thể
- Phân rã hệ thống thành các hệ thống con
- Mô tả các thành phần vật lý của hệ thống
- Bố trí các thành phần khả thi vào các nút phần cứng

Mục đích của thiết kế kiến trúc tổng thể

- Mục đích là **thiết kế kiến trúc tổng thể** của hệ thống
- Các thành phần tạo nên kiến trúc là gì phụ thuộc vào từng cách nhìn đối với hệ thống
- Kiến trúc tổng thể hệ thống có thể được nhìn theo 3 góc nhìn: Theo *hệ con*, Theo *thành phần phần mềm*, Theo *đơn vị phần cứng*
  - Phân rã hệ thống thành các hệ thống con (các gói)
    - Sơ đồ gói (Package diagram)
  - Mô tả các thành phần vật lý của hệ thống
    - Sơ đồ thành phần (Component diagram)
  - Bố trí các thành phần khả thi vào các nút phần cứng
    - Sơ đồ triển khai (Deployment diagram)

Phân rã hệ thống thành các hệ thống con (1)

- Khái niệm về hệ thống con (subsystem)
  - Các lớp là những thực thể cấu trúc rất nhỏ so với một hệ thống thực. Bởi vậy, khi số các lớp trong hệ thống đã lên tới hàng chục, ta nên gom các lớp liên quan với nhau thành từng nhóm gọi là các hệ thống con.
  - **Hệ thống con (subsystem)** là sự gom nhóm một cách logic (hợp lý) các lớp có sự **gắn kết mạnh bên trong** (của hệ thống con) và sự **liên kết yếu bên ngoài** (giữa các hệ thống con)
  - UML dùng thuật ngữ **gói (package)**, cho nên ta cũng sẽ biểu diễn hệ con dưới dạng gói, mang theo khuôn dập <<subsystem>>

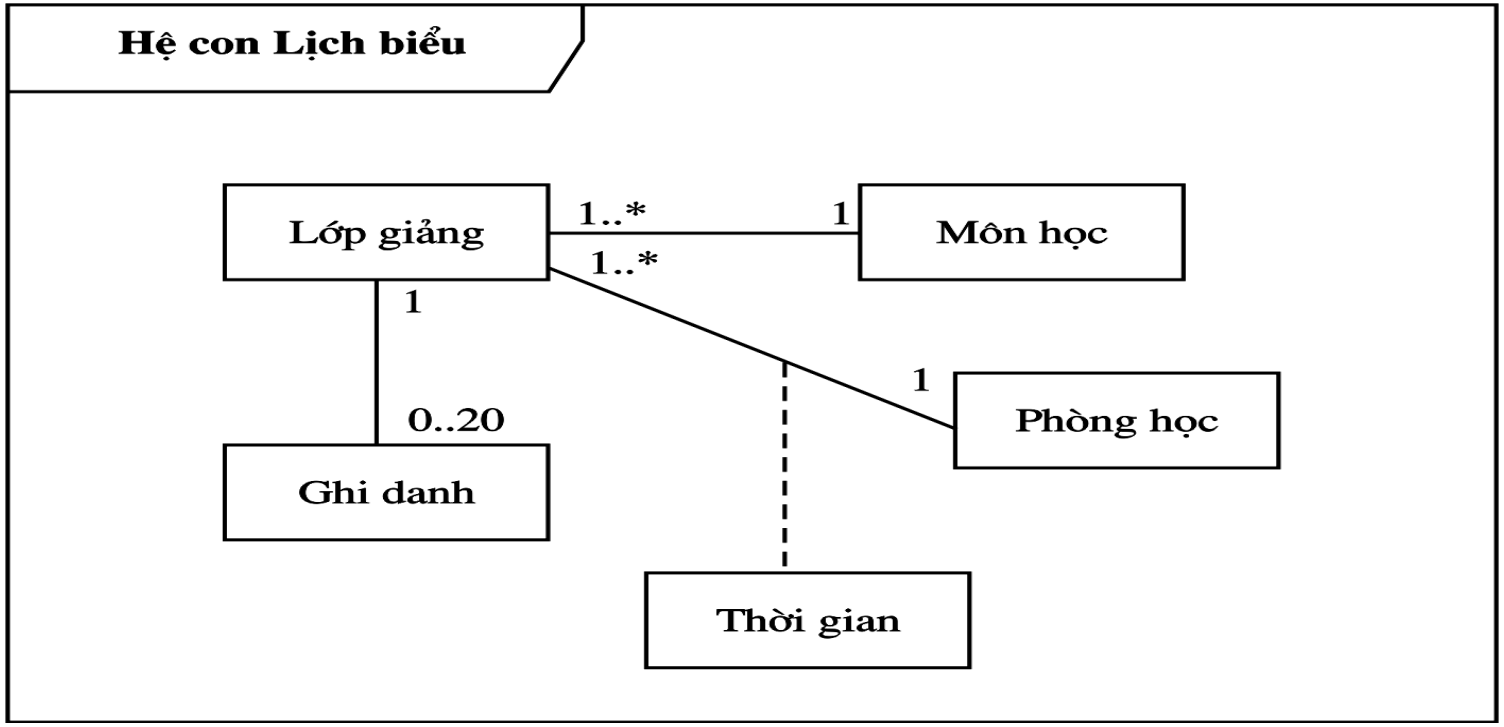
Phân rã hệ thống thành các hệ thống con (2)

- Nội dung của một hệ thống con (gồm các lớp và các mối liên quan giữa chúng) được UML 2.0 diễn tả trong một khung (frame), với một tựa đề viết trong một hình chữ nhật cắt góc theo khuôn dạng:

  [<loại>] Tên [<tham số>]

Phân rã hệ thống thành các hệ thống con (3)

Số các lớp trong một hệ thống con không nên ít quá hay nhiều quá



Phân rã hệ thống thành các hệ thống con (4)

Sự **gắn kết** mạnh của các lớp trong cùng một hệ thống con thể hiện:

- ***Về mục đích***: Các lớp phải cung cấp các dịch vụ có cùng bản chất cho người dùng. Như vậy chúng phải thuộc vào cùng một lĩnh vực và đề cập một số thuật ngữ chung (chẳng hạn hệ thống con giao diện đề cập các thuật ngữ như: cửa sổ, thực đơn, nút nhấn,...)
- ***Về xu thế phát triển***: Người ta tách các lớp bền vững với các lớp có nhiều khả năng thay đổi. Đặc biệt, thường tách các lớp nghiệp vụ với các lớp ứng dụng, và xếp chúng vào các hệ con khác nhau.
- ***Về ứng dụng các công nghệ***: Để tận dụng các công nghệ có sẵn, như các thư viện chương trình (lớp/thành phần), các GUI, các hệ quản trị cơ sở dữ liệu,…, ta thường tách các hệ thống con giao tiếp, hệ thống con quản trị dữ liệu ra khỏi phần lõi (ứng dụng và nghiệp vụ) của hệ thống

Phân rã hệ thống thành các hệ thống con (5)

- Sự **liên kết** giữa các hệ thống con thể hiện ở mối liên quan **phụ thuộc** giữa chúng
- Sự phụ thuộc giữa 2 hệ thống con phản ánh **các mối liên quan tĩnh** (thừa kế, liên kết, …) và **các mối liên quan động** (trao đổi thông điệp) giữa các lớp thuộc 2 hệ thống con đó
- Sự phụ thuộc giữa các hệ thống con phải càng đơn giản, lỏng lẻo thì càng tốt. Để đảm bảo tính liên kết yếu này, khi thành lập hệ thống con, áp dụng các quy tắc sau:
  - Các lớp thuộc vào cùng một cấu trúc thừa kế (inheritance hierarchy) nên được xếp vào cùng một hệ thống con
  - Các lớp có mối liên quan kết nhập và hợp thành với nhau thường được xếp vào cùng một hệ thống con
  - Các lớp cộng tác với nhau nhiều, trao đổi thông tin nhiều, thể hiện qua các biểu đồ tương tác, thì nên đặt chung vào một hệ thống con
  - Nên tránh sự phụ thuộc vòng quanh giữa các hệ thống con

Phân rã hệ thống thành các hệ thống con (6)

- Kiến trúc phân tầng
  - Một hệ thống con thường được định nghĩa bởi các dịch vụ mà nó cung cấp
    - Mối liên quan giữa một hệ thống con với phần còn lại của hệ thống có thể là ngang hàng hay là khách/chủ
  - Trong mối **liên quan ngang hàng (peer-to-peer)** thì mỗi bên đều có thể truy cập các dịch vụ của bên kia
  - Còn mối **liên quan khách/chủ (client/server)** thì đơn giản hơn: bên khách (client) gọi bên chủ (server) và bên chủ thực hiện một dịch vụ theo yêu cầu và trả kết quả cho bên khách
    - Bên khách phải biết giao diện của bên chủ
    - Nhưng bên chủ thì không cần biết giao diện của bên khách

Phân rã hệ thống thành các hệ thống con (7)

- Từ 2 hình thức giao tiếp đó, ta có hai cách để chia hệ thống thành các hệ thống con:
  - Tổ chức hệ thống thành các **tầng**, với mối quan hệ khách/chủ luôn luôn hướng từ tầng trên xuống (các) tầng dưới
    - Tầng trên đóng vai trò khách, tầng dưới đóng vai trò chủ
    - Ví dụ: Hệ thống tạo cửa sổ trong giao diện người dùng của máy tính
  - Tổ chức hệ thống thành các **lát**, với mối quan hệ ngang hàng giữa các lát; tuy nhiên các lát là khá độc lập hoặc liên kết yếu với nhau
    - Ví dụ: Hệ điều hành thường gồm các hệ con như là các hệ quản lý tệp, hệ điều khiển thiết bị, hệ quản lý sự kiện và ngắt, …

Phân rã hệ thống thành các hệ thống con (8)

- Tổ chức phân tầng là đáng được ưu tiên hơn, vì nó mang lại nhiều ưu thế trong thiết kế, trong cài đặt, cũng như trong việc sử dụng lại
- **Đối với các hệ thống lớn, thì ta thường phải phối hợp cả 2 cách tổ chức phân tầng và phân lát**
  - Phân hệ thống thành các tầng (architectural building layers)
  - Trong mỗi tầng, thì lại phân thành các lát (architectural building blocks)

Phân rã hệ thống thành các hệ thống con (9)

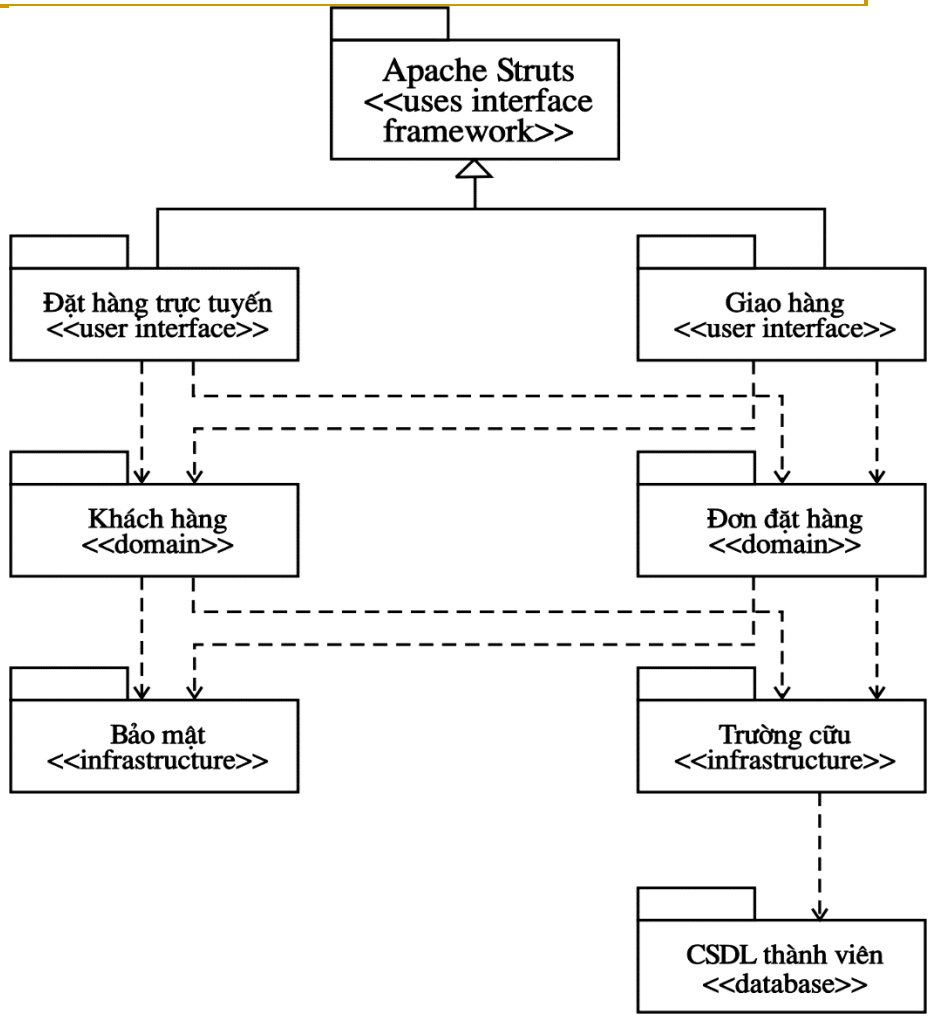
- Khi thực hiện phân tầng, thì số tầng là tuỳ thuộc sự phức tạp của hệ thống:
  - Trong một hệ thống đơn giản, thì số tầng có thể chỉ là hai (2 tiers): Tầng khách (client) thì quản lý giao diện người dùng và các quá trình khai thác, còn tầng máy chủ (server) thì xử lý việc lưu giữ các dữ liệu
  - Trong một hệ thống phức tạp hơn, thì người ta tách tầng trên thành 2 tầng: *giao diện* và *ứng dụng*, và ở dưới nó là tầng *nghiệp vụ* (hay tầng lĩnh vực), bền vững hơn và có nhiều khả năng sử dụng lại hơn
    - Đó là một kiến trúc khách/chủ ba tầng (3 tiers)

Phân rã hệ thống thành các hệ thống con (10)

- Trong các hệ thống lớn, số tầng còn có thể nhiều hơn (n tiers)
- Điển hình là kiến trúc 5 tầng (từ trên xuống):
  - **Tầng trình bày:** Chuyển các dữ liệu cho người dùng và biến đổi các hành động của người dùng thành các sự kiện vào của hệ thống
  - **Tầng ứng dụng:** Bao gồm các đối tượng điều khiển và dẫn dắt các quy luật của ứng dụng
  - **Tầng nghiệp vụ:** Bao gồm các đối tượng nghiệp vụ (hay lĩnh vực), và việc cài đặt các quy tắc quản lý chúng
  - **Tầng truy cập dữ liệu:** Quản lý việc truy cập (đọc/viết) các đối tượng nghiệp vụ từ các phương tiện lưu trữ dữ liệu
  - **Tầng lưu trữ dữ liệu:** Bảo đảm sự lưu giữ lâu dài các dữ liệu

Phân rã hệ thống thành các hệ thống con (11)

- Hình vẽ bên minh hoạ về kiến trúc khách/chủ gồm 5 tầng, trong đó mỗi gói (hệ thống con) đều có mang khuôn dập thích hợp:
  - <<user interface framework>>: khung giao diện người dùng
  - <<user interface>>: giao diện người dùng
  - <<domain>>: lĩnh vực
  - <<infrastructure>>: cơ sở hạ tầng
  - <<database>>: cơ sở dữ liệu



Mô tả các thành phần vật lý của hệ thống (1)

- Thành phần (Components) và biểu đồ thành phần (Component diagram)
  - Nếu như biểu đồ gói (hệ thống con) mà ta nói ở phần trên phản ánh cho góc nhìn về cấu trúc logic của hệ thống (ở mức cao so với biểu đồ lớp), thì biểu đồ thành phần cho ta một cách nhìn về **cấu trúc vật lý của hệ thống**
  - Chữ "vật lý" ở đây được hiểu theo nghĩa là sự mô tả hướng tới các sản phẩm phần mềm, là kết quả của sự cài đặt và thực sự tồn tại
    - Chứ không phải là các sản phẩm logic, là kết quả của quá trình phân tích
  - Tuy nhiên, ở đây ta chưa đề cập tới phần cứng, mặc dù tính vật lý của nó cũng là đương nhiên

Mô tả các thành phần vật lý của hệ thống (2)

- UML định nghĩa **thành phần (component)** là một bộ phận vật lý và thay thế được của hệ thống, cung cấp sự thực hiện (implementation) cho một tập các giao diện (interfaces)
- Nói cách khác, thì thành phần là một cài đặt của một tập hợp các phần tử logic (vd: các lớp hay các hợp tác)

Mô tả các thành phần vật lý của hệ thống (3)

- Có 3 loại thành phần:
  - Các **thành phần triển khai (deployment components)**: Đó là các thành phần cần và đủ để tạo nên một hệ thống khả thi, như là các thư viện động (DLL) và các mã thực thi (executable). Định nghĩa thành phần của UML là đủ rộng để bao hàm các mô hình đối tượng kinh điển (vd: COM+, CORBA, và EJB - Enterprise Java Beans), cũng như các mô hình đối tượng khác như là các trang Web động, các bảng cơ sở dữ liệu, và các mã thực thi sử dụng những cơ chế truyền thông riêng

Mô tả các thành phần vật lý của hệ thống (4)

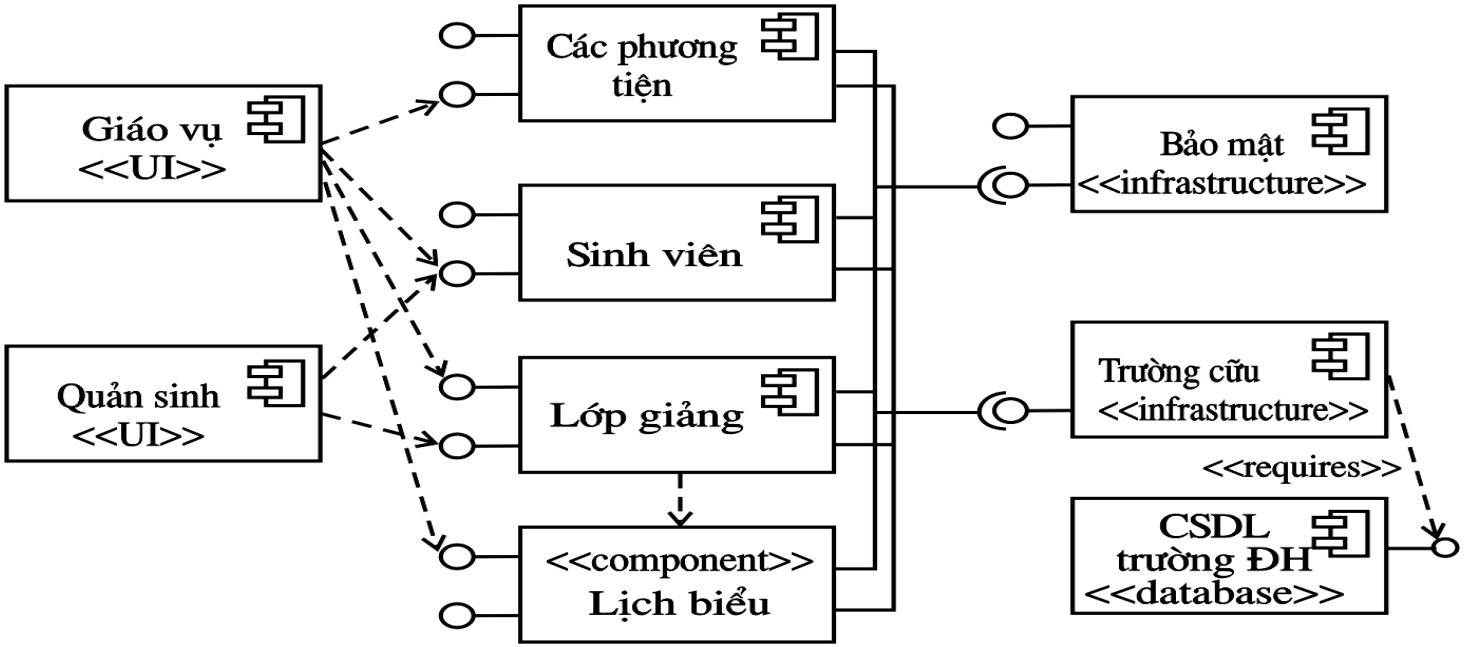
- Các **thành phần sản phẩm làm việc (work product components)**: Đó là các thành phần có từ quá trình phát triển hệ thống, bao gồm các tệp mã nguồn, các tệp dữ liệu, từ đó mà ta đã tạo lập ra các thành phần triển khai. Các thành phần này không trực tiếp tham gia vào hệ thống thực thi, nhưng không có chúng thì không tạo được hệ thống thực thi
- Các **thành phần thực hiện (execution components)**: Đó là các thành phần được tạo nên như là một kết quả của một hệ thực hiện (vd: một thành phần .JAR).

Mô tả các thành phần vật lý của hệ thống (5)

- 5 khuôn dập chuẩn của UML
  - <<executable>>: Một thành phần có thể thực hiện trên 1 nút
  - <<library>>: Một thư viện đối tượng tĩnh hoặc động
  - <<table>>: Một bảng trong cơ sở dữ liệu
  - <<file>>: Một tập tin chứa mã nguồn hoặc dữ liệu
  - <<document>>: Một tài liệu
- Có thể dùng các khuôn dập không chuẩn
  - <<application>>: Một ứng dụng
  - <<database>>: Một cơ sở dữ liệu
  - <<infrastructure>>: Một thành phần cơ sở hạ tầng (vd: một dịch vụ lưu trữ, một kiểm soát đăng nhập,…)
  - <<source code>>: Một tập tin mã nguồn
  - <<web service>>: Một dịch vụ Web

Mô tả các thành phần vật lý của hệ thống (6)

- Để tổ chức các thành phần lại với nhau, ta có hai cách:
  - Gom các thành phần vào các gói, nghĩa là đưa chúng vào các hệ thống con; hoặc
  - Thiết lập các mối liên quan phụ thuộc giữa chúng, và như thế ta có một Biểu đồ thành phần



Mô tả các thành phần vật lý của hệ thống (7)

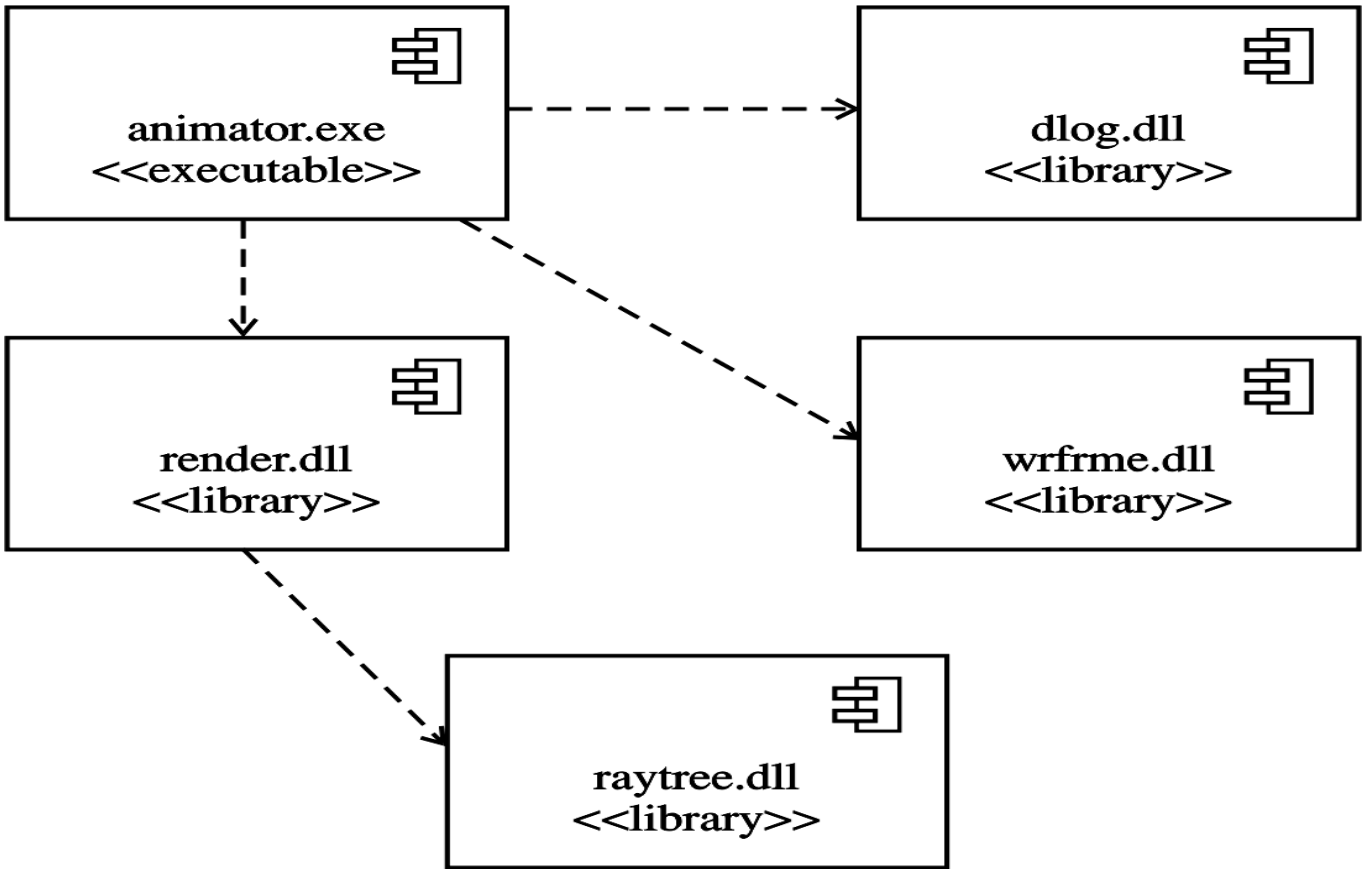
- Các mục đích mô hình hoá của Biểu đồ thành phần
  - Như đã trình bày, có nhiều loại thành phần: thành phần triển khai, thành phần sản phẩm làm việc, và thành phần thực hiện
  - Vì vậy, Biểu đồ thành phần lập ra phải có mục đích ***mô tả loại thành phần nào***

Mô tả các thành phần vật lý của hệ thống (8)

- **Mô hình hoá các thành phần khả thi và thư viện**: Mục đích chính của việc sử dụng Biểu đồ thành phần là để mô hình hoá các thành phần triển khai, tạo nên sự cài đặt của hệ thống
  - Nếu ta muốn cài đặt một hệ thống chỉ gồm đúng một tệp khả thi (.EXE), thì ta chẳng cần dùng tới thành phần
  - Nhưng nếu hệ thống gồm nhiều tệp khả thi và liên kết với các thư viện đối tượng, thì ta cần dùng biểu đồ thành phần để giúp hiển thị, đặc tả, thành lập và tư liệu hoá các quyết định đối với hệ thống vật lý

Mô tả các thành phần vật lý của hệ thống (9)

Ví dụ: Một biểu đồ các thành phần triển khai

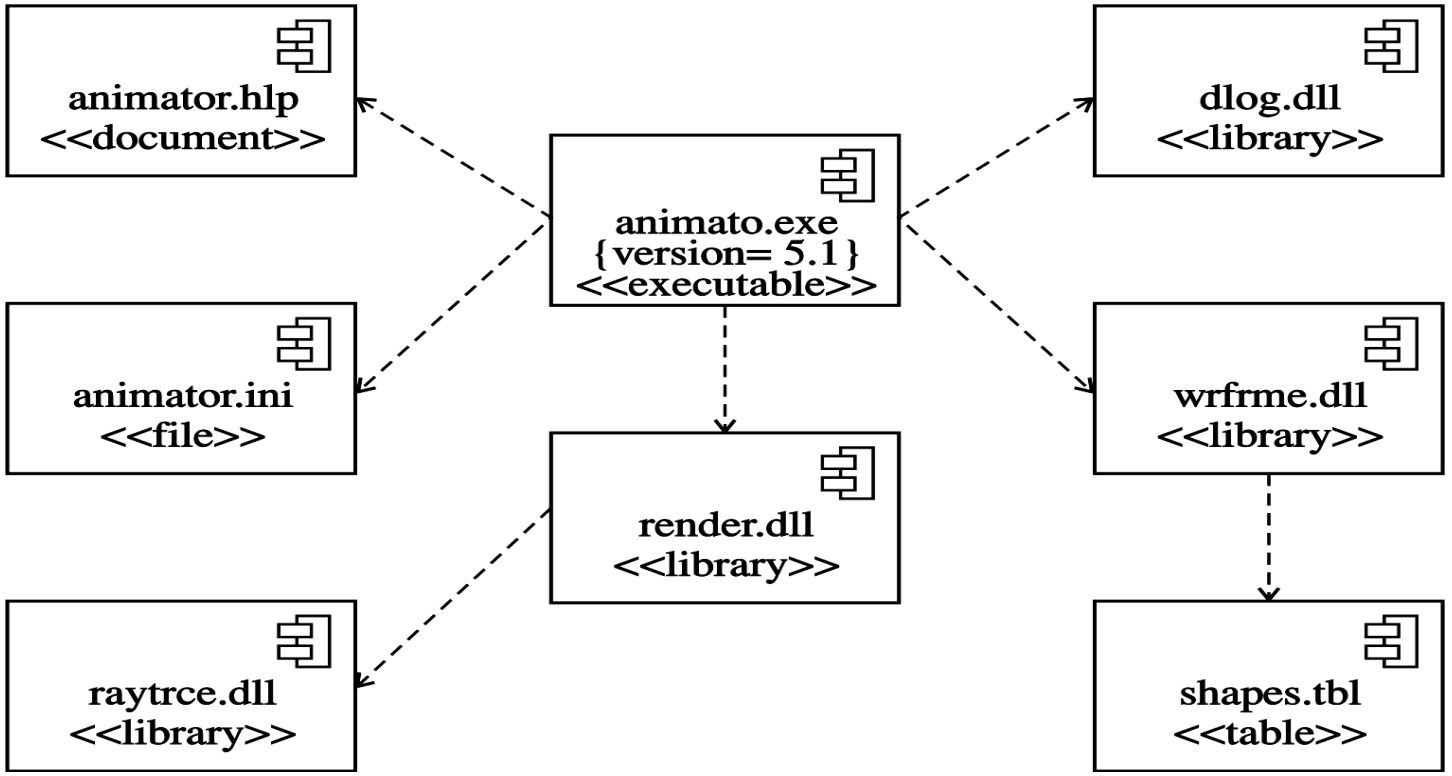


Mô tả các thành phần vật lý của hệ thống (10)

- **Mô hình hoá các bảng, các tệp và các tài liệu**
  - Bên cạnh các tệp khả thi và thư viện tạo nên phần chạy được của hệ thống, thì còn nhiều thành phần bố trí khác, gọi là các thành phần phụ trợ, cần cho việc cài đặt hệ thống
  - Ví dụ: Để cài đặt, ta vẫn cần các tệp dữ liệu, các tài liệu trợ giúp, các scripts, các tệp log, các tệp khởi tạo tham số cấu hình,…
  - Mô hình hoá các thành phần này cũng là một phần quan trọng để diễn tả hình trạng của hệ thống

Mô tả các thành phần vật lý của hệ thống (11)

Ví dụ: Từ biểu đồ ở Slide 25, thêm vào các bảng dữ liệu, các tệp và các tài liệu

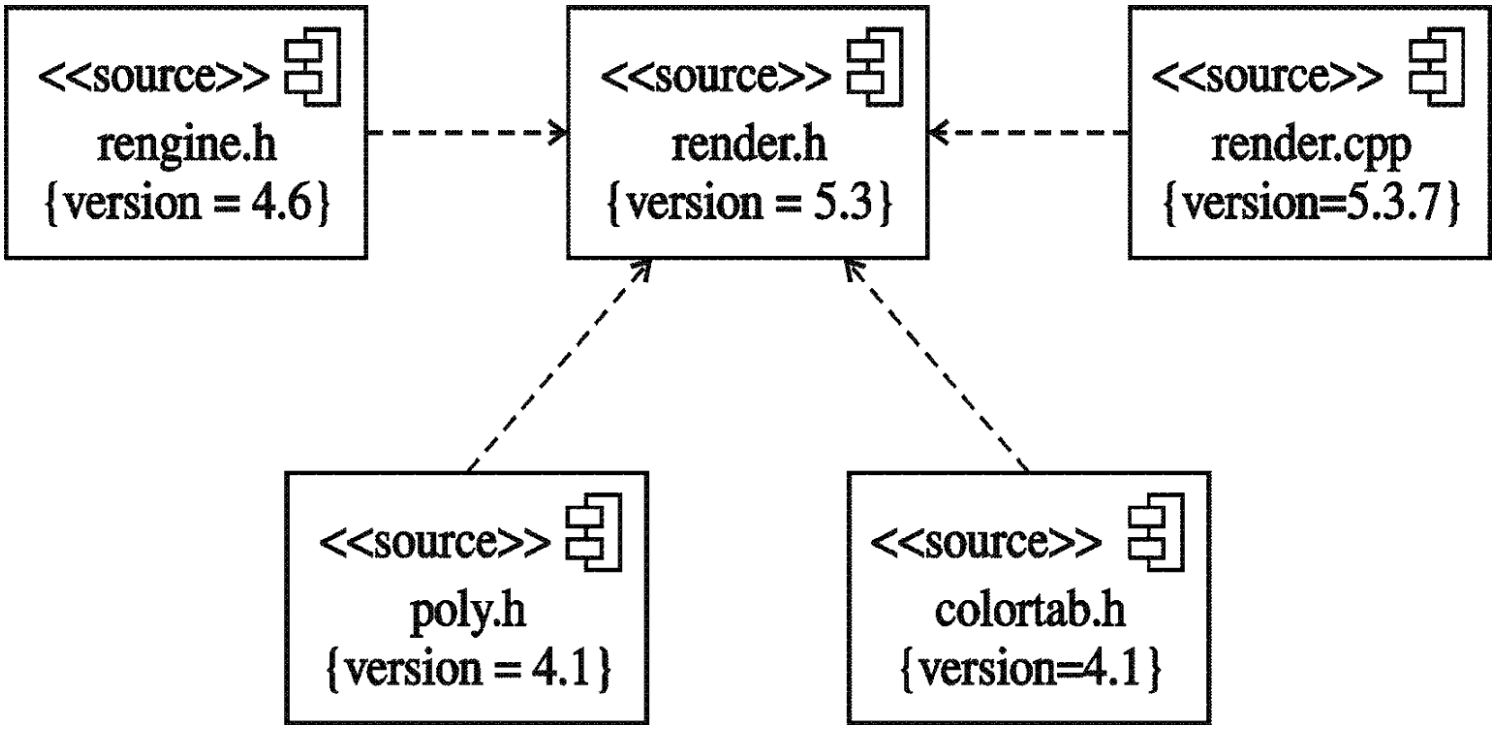


Mô tả các thành phần vật lý của hệ thống (12)

- **Mô hình hoá mã nguồn**
  - Mô hình hoá mã nguồn cũng là một mục đích của việc tạo lập biểu đồ thành phần
  - Các tệp mã nguồn dùng để chứa các chi tiết về các lớp, các giao diện, các hợp tác và các phần tử logic khác
  - Các tệp mã nguồn tạo nên một bước trung gian để tạo lập các thành phần vật lý/thực thi
  - Các công cụ (vd: chương trình biên dịch) thường đòi hỏi các tệp mã nguồn phải được tổ chức theo một quy cách nhất định nào đó (vd: tệp header .h và tệp mã nguồn .cpp). Các mối liên quan phụ thuộc giữa các tệp này phản ánh một trật tự biên dịch

Mô tả các thành phần vật lý của hệ thống (13)

Ví dụ: Mô hình hoá các tệp mã nguồn dùng để xây dựng thư viện render.dll từ ví dụ trước



Bố trí các thành phần khả thi vào các nút phần cứng (1)

- Để bố trí các thành phần phần mềm vào các nút phần cứng, ta dùng Biểu đồ triển khai
- **Biểu đồ triển khai (Deployment diagram)** là một biểu đồ diễn tả sự bố trí các đối tượng thực thi (executable artifacts) trên nền tảng thực thi (underlying platform). Biểu đồ triển khai gồm các nút và các kết nối giữa các nút đó
- Một **nút (node)** là một phần tử vật lý tồn tại vào lúc chạy và biểu diễn cho một **tài nguyên tính toán (computational resource)**. Một nút được biểu diễn bởi một hình hộp, có mang tên.
  - Nếu tên này không gạch dưới, thì nút thể hiện một **lớp các tài nguyên**
  - Nếu tên được gạch dưới, thì nút thể hiện một **cá thể tài nguyên**

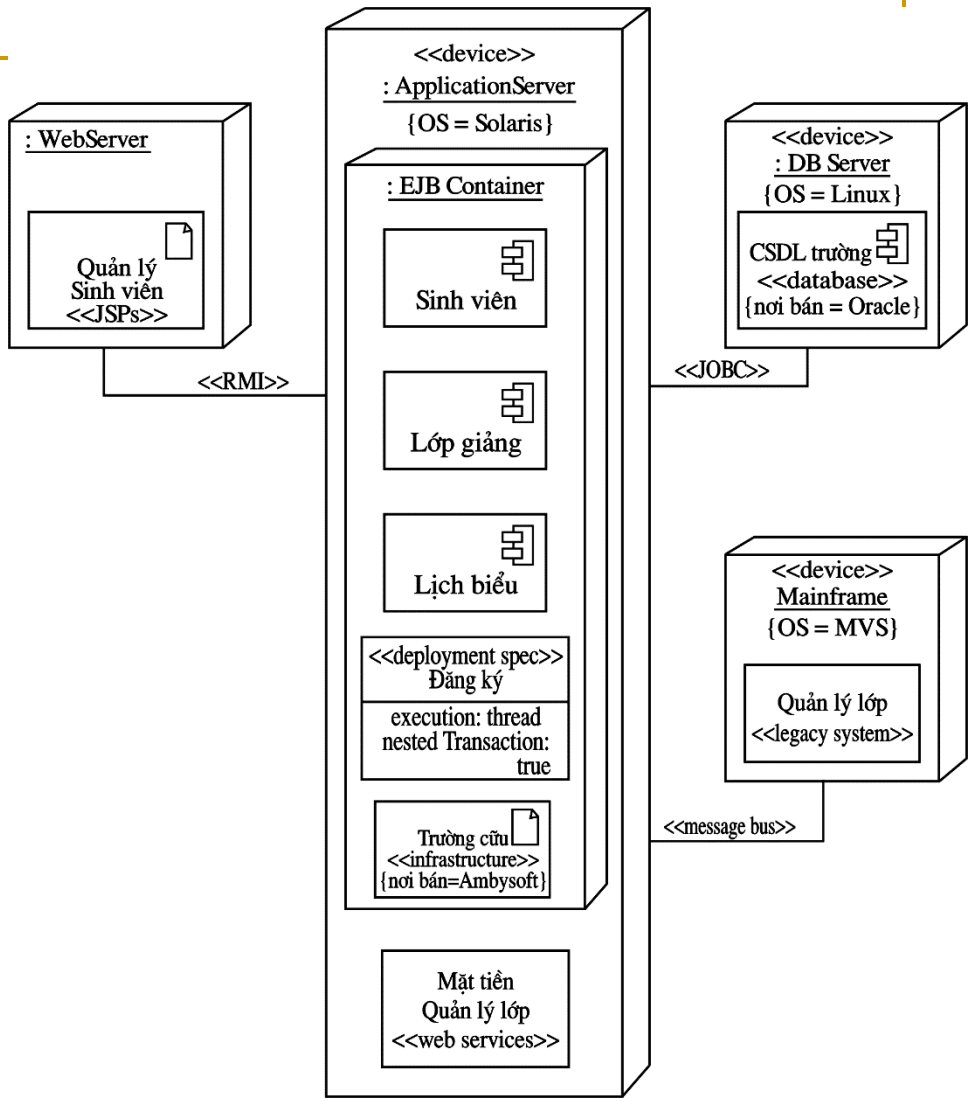
Bố trí các thành phần khả thi vào các nút phần cứng (2)

- Có hai mức diễn tả của Biểu đồ triển khai: mức lớp (tương ứng với Biểu đồ lớp) và mức cá thể (tương ứng với Biểu đồ đối tượng)
- Nút có thể là:
  - Một thiết bị (nút phần cứng), thường mang khuôn dập <<device>> (hoặc cụ thể hơn là <<processor>>, <<console>>, <<kiosk>>, <<printer>>,…)
  - Một môi trường thực hiện (nút phần mềm), thường mang khuôn dập <<execution env>> (vd: <<EJB Container>>, <<J2EEServer>>)

Bố trí các thành phần khả thi vào các nút phần cứng (3)

Hình vẽ bên thể hiện một biểu đồ triển khai (ở mức cá thể) biểu diễn cho hình trạng vật lý của Hệ thống thông tin của 1 trường đại học

- Nút **WebServer** chưa có khuôn dập, đó là vì người phát triển hệ thống chưa có quyết định – Nút này có thể là một loại phần mềm hoặc là một thiết bị vật lý
- Các nút có thể chứa các nút khác hoặc các phần mềm. Chẳng hạn, nút **ApplicationServer** chứa nút **EJBContainer** (một nút phần mềm), và nút này lại chứa ba thành phần phần mềm, một đặc tả bố trí, và một phần mềm



Bố trí các thành phần khả thi vào các nút phần cứng (4)

- Các **kết nối (connections)** là các mối liên quan giao tiếp giữa các cặp nút, thể hiện về mặt vật lý bằng một đường truyền (vd: một kết nối Ethernet, một đường truyền tuần tự, hay một kênh truyền dùng chung)
- Mỗi kết nối hỗ trợ cho một hay nhiều giao thức truyền thông, mà ta cần chỉ rõ bằng các khuôn dập thích hợp

Bố trí các thành phần khả thi vào các nút phần cứng (5)

Một số khuôn dập dùng cho các kết nối

Khuôn dập	**Ý nghĩa**
Asynchronous	Một kết nối không đồng bộ,
HTTP	HyperText Transport Protocol, một giao thức Internet
JDBC	Java Database Connectivity, công nghệ Java để kết nối và truy cập CSDL
ODBC	Open Database Connectivity, kết nối và truy cập CSDL
RMI	Remote Method Invocation, một giao thức hỗ trợ lời gọi từ xa
RPC	Remote Procedure Call, lời gọi thủ tục từ xa
Synchronous	Một kết nối đồng bộ, trong đó bên gửi chờ trả lời từ bên nhận
Web services	Liên lạc qua các giao thức web services (vd: SOAP)

Thiết kế giao diện người dùng

Mục đích

Mô tả các giao diện của hệ thống

Làm nguyên mẫu

Mục đích của làm nguyên mẫu giao diện người dùng

- Dựa vào các công cụ tạo lập giao diện người dùng (GUI – Graphical user interface – design tools) ta thành lập sớm và nhanh một nguyên mẫu giao diện người dùng (GUI prototype), có tính thăm dò sự phù hợp, nhằm các mục đích:
  - Tạo ra một môi trường làm việc cụ thể, dễ tiếp xúc, dễ làm thử, làm cho người dùng trở nên yên tâm hơn, và năng động hơn trong việc đóng góp cho việc phát triển hệ thống
  - Qua quá trình dùng thử, ta thu thập được nhiều ý kiến phản hồi có ích từ phía người dùng
  - Sớm phát hiện được các yêu cầu hay chức năng bị bỏ sót, sớm nhìn thấy các điểm yếu, chỗ khó khăn nhất của hệ thống
- **Thiết kế giao diện không phải là các ảnh chụp màn hình hệ thống sau cài đặt!**

Mô tả các giao diện của hệ thống (1)

- Như đã biết, cứ mỗi cặp (**tác nhân, ca sử dụng**) liên quan, có ít nhất một lớp biên để chuyển đổi các thông tin vào/ra
  - Thể hiện của lớp biên chính là giao diện mà bây giờ ta cần phải mô tả
- Muốn mô tả các lớp biên, ta xem lại từng bước trong kịch bản của mỗi ca sử dụng – để xác định các phần tử của giao diện:
  - Xét nội dung của tương tác giữa tác nhân và hệ thống
  - Các thông tin vào và Các thông tin ra
  - Các hành động được yêu cầu

Mô tả các giao diện của hệ thống (2)

Bản mô tả một giao diện thường chứa các điểm sau:

- Tên của giao diện;
- Mô tả tóm tắt nội dung của giao diện;
- Mức độ phức tạp của giao diện (phức tạp/chuẩn/đơn giản);
- Ghi chú thêm (nếu có);
- Thiết kế chi tiết của giao diện, tuỳ thuộc vào 4 loại giao diện sau:
  - Các giao diện hội thoại (với người dùng)
  - Các giao diện hiển thị thông tin (kết quả xử lý, báo cáo,…)
  - Các giao diện dữ liệu từ/đến các hệ thống ngoài
  - Các giao diện chức năng đến các hệ thống ngoài

Làm nguyên mẫu giao diện (1)

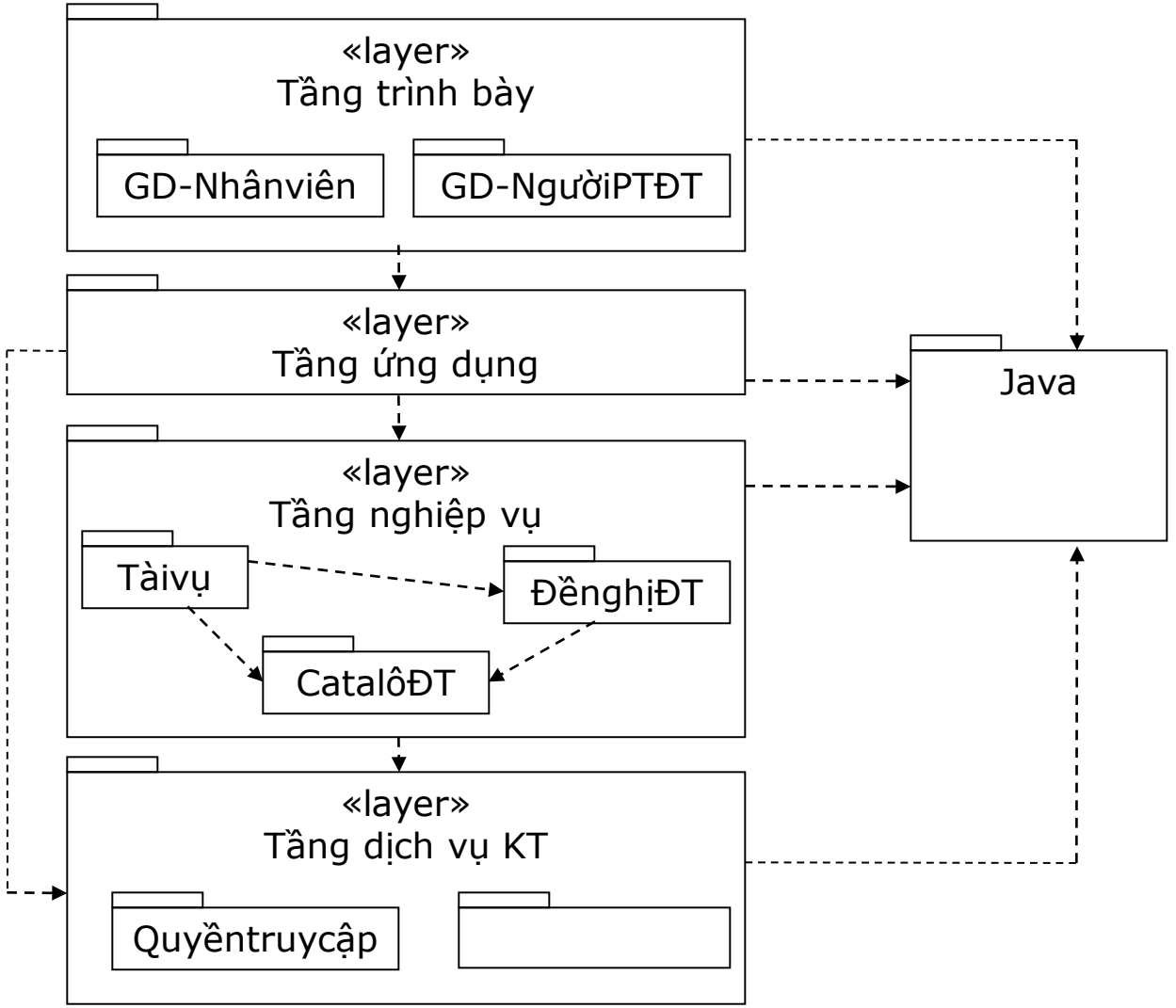
- Làm nguyên mẫu giao diện nên được bắt đầu càng sớm càng tốt
  - Có thể bắt đầu làm nguyên mẫu giao diện ngay khi xong phân tích các ca sử dụng
- Có nhiều bộ tạo lập giao diện người dùng (GUI builders) cho phép làm các nguyên mẫu giao diện mà không tốn mấy công sức
- **Đối với phiên bản đầu của nguyên mẫu giao diện, thì các trường là rỗng hoặc mang giá trị minh họa, các nút và các phần tử tương tác có thể chưa cần có hiệu ứng tương tác rõ ràng**
- **Qua các vòng lặp tiếp theo (của việc làm nguyên mẫu), thì giao diện dần trở nên sinh động hơn (dần đi tới phương án cuối)**
- Như vậy, người dùng có thể sớm làm việc thử với các nguyên mẫu giao diện

Làm nguyên mẫu giao diện (2)

- Nguyên mẫu giao diện chỉ có ý nghĩa thăm dò => nên làm nhanh và không nên cầu toàn
  - Chưa nên chú ý nhiều về trình bày và mỹ thuật, mà cần chú ý đến ***nội dung*** (các trường và các điều khiển của giao diện) và ***luồng dẫn dắt*** *từ phần tử giao diện này sang phần tử giao diện khác* (screens flow)
- Quá trình phát triển nguyên mẫu giao diện nên được làm song song với quá trình phân tích và thiết kế, và nên hỗ trợ cho phân tích và thiết kế
- Nguyên mẫu giao diện nói chung chỉ là mặt ngoài của hệ thống, không phản ánh hết tầm sâu (các chi tiết hoạt động) của hệ thống
  - Cần nói rõ với người dùng là nguyên mẫu không phải là hệ thống hoàn chỉnh
- **Làm nguyên mẫu chỉ là một sự hỗ trợ tốt, chứ không thể là sự thay thế cho các bước phân tích và thiết kế hệ thống một cách nghiêm túc được**

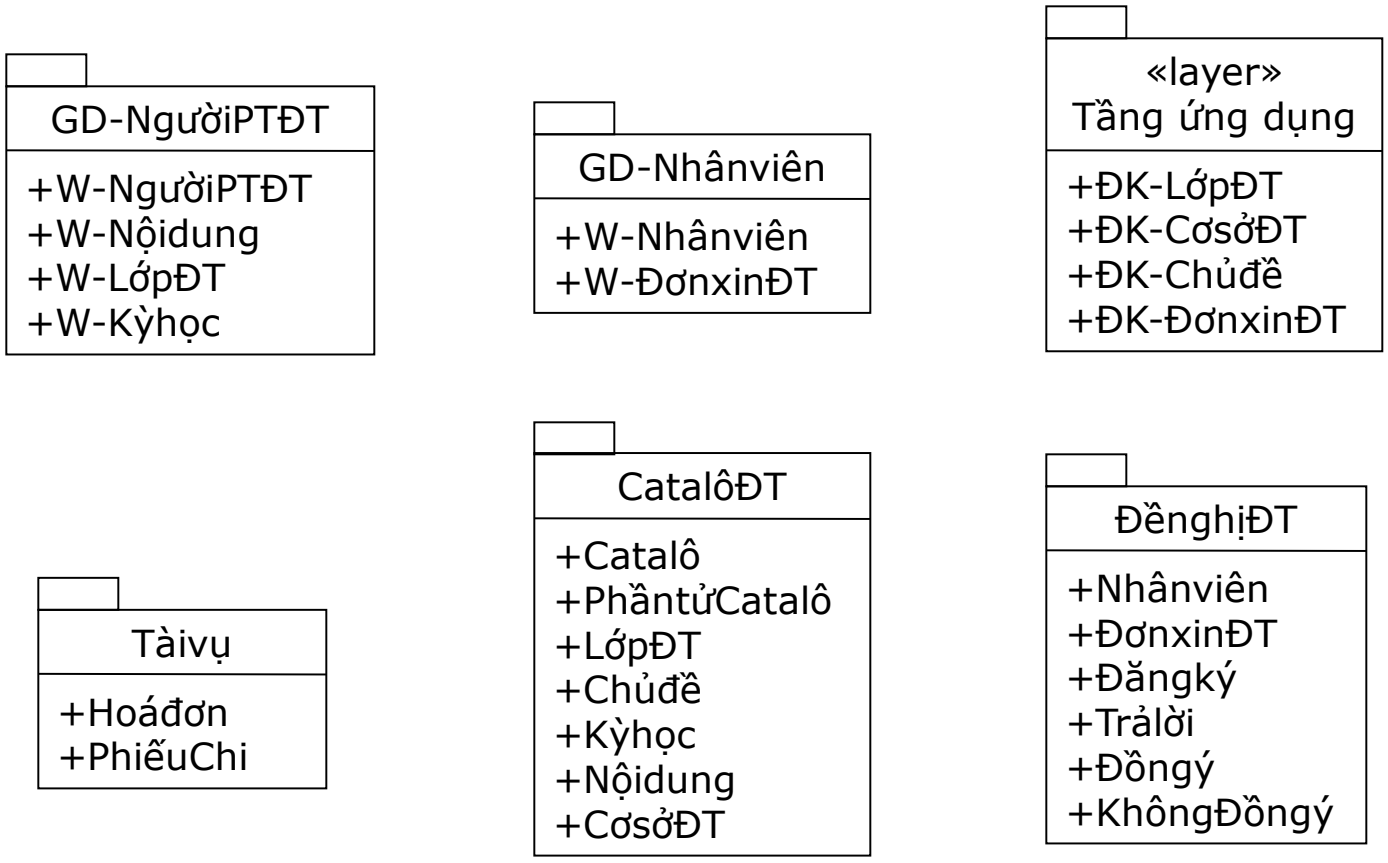
Bài tập tổng hợp (1)

Đề xuất một Kiến trúc phân tầng cho Hệ thống yêu cầu đào tạo



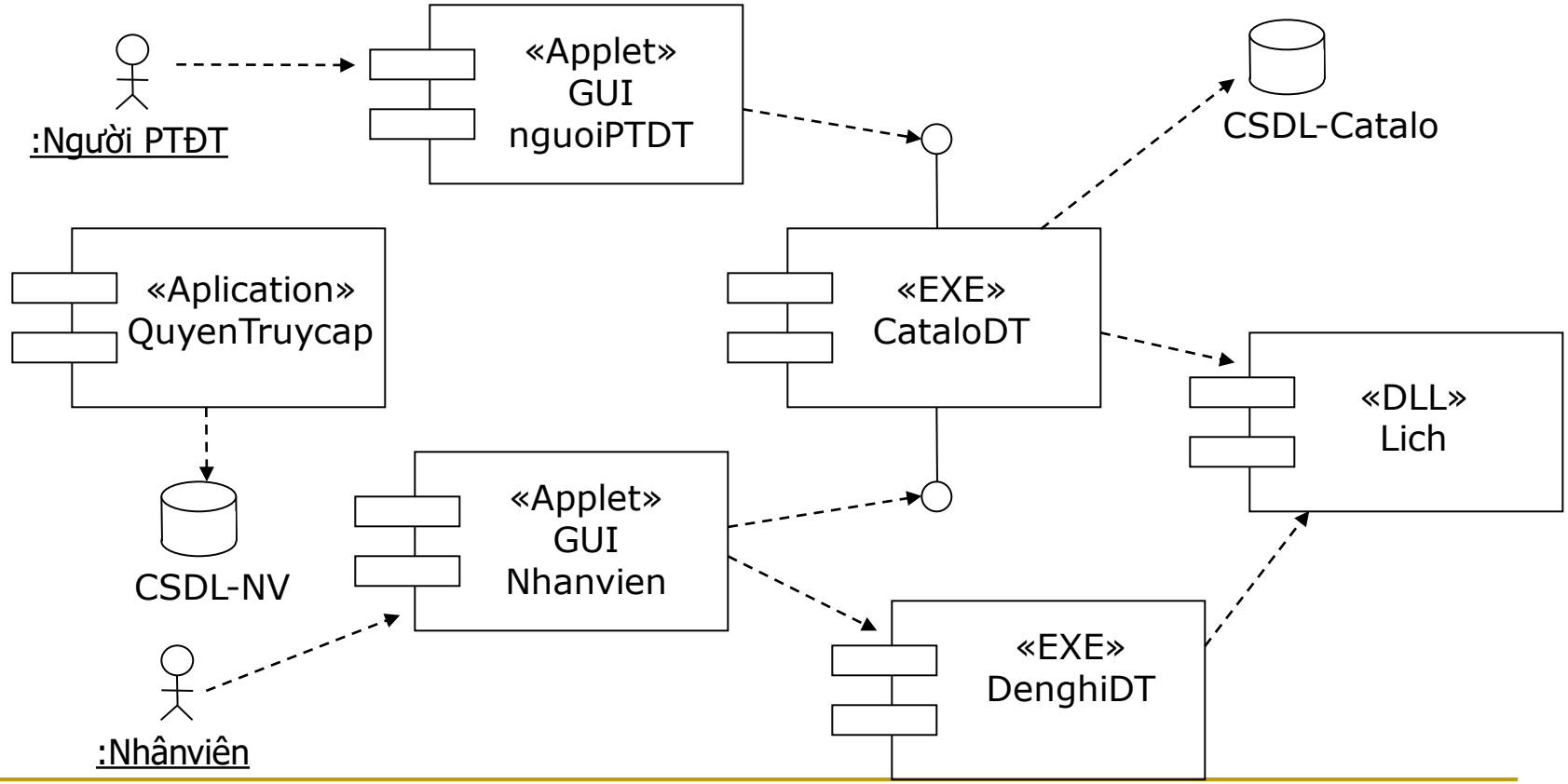
Bài tập tổng hợp (2)

Trong kiến trúc phân tầng trên, thì gói Java chứa các lớp cơ sở dùng chung cho mọi tầng, còn các gói khác thì chứa các lớp như sau:



Bài tập tổng hợp (3)

Câu hỏi 27: *Hãy đề xuất một Biểu đồ thành phần của hệ thống. Chỉ cần đưa ra các thành phần đủ để biểu diễn cho một hợp tác giữa nhiều lớp. Ngôn ngữ đích là Java.*



Bài tập tổng hợp (4)

Câu hỏi 28: *Hãy đề xuất một Biểu đồ triển khai của hệ thống*.