**1: 2: 3:**

**4:**

Trong phần này, nhóm sẽ mô tả những gì được thực hiện trong Xác định các phần tử thiết kế, nhưng sẽ không mô tả làm như thế nào. Vì nó là một môn học blah...blah...

//Hiểu được lý do và cân nhắc hỗ trợ các quyết định kiến trúc là cần thiết để hiểu trong đó thiết kế được phát triển.

**5:**

Các kiến trúc sư thực hiện Xác định các phần tử thiết kế

* Mục đích:

Để phân tích mối tương tác giữa các lớp phân tích để xác định các yếu tố Mô hình thiết kế

* Input:

+ Bổ sung thông số kỹ thuật

+ Tài liệu kiến trúc phần mềm

+ Dự án hướng dẫn cụ thể

+ Lớp phân tích

+ Phân tích mẫu

+ Mô hình thiết kế

* Kết quả:

+ Phần tử thiết kế mẫu

Lớp

Gói

Hệ thống con

**//6:**

Xác định các phần tử thiết kế là nơi các lớp phân tích được xác định trong Use-Case. Phân tích được chuyển thành các yếu tố thiết kế ( các lớp, các hệ thống con ). Lớp phân tích xử lý các yêu cầu chủ yếu là chức năng, và đối tượng mô hình từ “ vấn đề”, lĩnh vực; phần tử thiết kế xử lý yêu cầu không có chức năng và mô hình các đối tượng từ các “giải pháp” tên miền.

Nó nằm trong XĐCPTK khi quyết định lớp phân tích nào là một lớp mà các hệ thống con (phải được phân hủy tiếp tục), hiện có và không cần “thiết kế” tất cả.

Một khi các lớp thiết kế và hệ thống con được tạo ra , phải giao cho một tên và một mô tả ngắn. Trách nhiệm của các lớp phân tích ban đầu nên được chuyển sang các hệ thống con mới được tạo ra. Ngoài ra, cơ chế thiết kế xác định cần gắn với các phần tử thiết kế.

**7:**

Nếu lớp phân tích đơn giản và đã được đại diện cho một khái niệm trừu tượng logic duy nhất, sau đó nó có thể được ánh xạ trực tiếp một – một để thành một lớp thiết kế. Thông thường, các lớp thực thể tồn tại tương đối nguyên vẹn vào thiết kế.

Trong suốt các hoạt động thiết kế, các lớp phân tích được chuyển thành các phần tử thiết kế ( các lớp , gói, hệ thống con). Một số lớp phân tích có thể được phân chia, tham gia, loại bỏ, hoặc không cần thao tác. Nói chung, có 1 ánh xạ nhiều-nhiều giữa các lớp phân tích và các phần tử thiết kế. Các ánh xạ bao gồm những điều sau:

* Một lớp phân tích có thể trở thành:

+ Một lớp duy nhất trong Mô hình thiết kế.

+ Một phần của một lớp trong Mô hình thiết kế.

//+ Một lớp tổng hợp trong mô hình thiết kế (có nghĩa là các bộ phận trong tổng hợp này có thể không được mô hình một cách rõ ràng trong mô hình phân tích).

//+ Một nhóm các lớp kế thừa từ lớp cùng trong Mô hình thiết kế.

+ Một nhóm các lớp liên quan đến chức năng trong Mô hình thiết kế (VD: gói)

+ Một hệ thống con trong Mô hình thiết kế.

//+ Một mối quan hệ trong mô hình thiết kế.

* Một mối quan hệ giữa các lớp phân tích có thể trở thành một lớp trong Mô hình thiết kế.
* //Một phần của một lớp phân tích có thể được thực hiện bởi phần cứng, và không phải mô hình trong Mô hình thiết kế ở tất cả.

**8:**

Khi xác định các lớp, nên nhóm chúng thành các gói cho các mục đích tổ chức và quản lý cấu hình.

Mô hình thiết kế có thể được cấu trúc thành các đơn vị nhỏ hơn để làm cho nó dễ dàng, dễ hiểu hơn. Bằng cách nhóm các yếu tố Mô hình thiết kế thành các gói và các hệ thống con, sau đó cho thấy làm thế nào những nhóm liên quan đến nhau để dễ hiểu hơn về cấu trúc tổng thể của mô hình.

Phân vùng Mô hình thiết kế 1 số lý do(yếu tố):

* //Có thể sử dụng gói phần mềm và hệ thống con là đơn vị đặt hàng, cấu hình hoặc giao hàng khi một hệ thống kết thúc.
* Phân bổ nguồn lực và năng lực của đội ngũ phát triển khác nhau có thể yêu cầu dự án được phân chia giữa các nhóm khác nhau tại các địa điểm khác nhau.
* Hệ thống con có thể được dùng để cấu trúc Mô hình thiết kế phản ánh các loại người dùng. Nhiều yêu cầu thay đổi bắt nguồn từ người sử dụng , hệ thống con đảm bảo rằng những thay đổi từ một loại người dùng cụ thể sẽ chỉ ảnh hưởng đến các bộ phận của hệ thống tương ứng với loại người dùng.
* Hệ thống con được sử dụng để đại diện cho các sản phẩm và dịch vụ hiện có mà hệ thống sử dụng.

**9:**

Khi các lớp ranh giới được phân phối cho các gói, có hai chiến lược khác nhau có thể được áp dụng. Mà một trong những lựa chọn phụ thuộc vào việc có hay không các giao diện hệ thống có thể sẽ thay đổi đáng kể trong tương lai.

Nếu có khả năng là các giao diện hệ thống sẽ được thay thế, hoặc trải qua những thay đổi đáng kể, giao diện phải được tách ra khỏi phần còn lại của thiết kế mẫu. Khi giao diện người dùng được thay đổi, chỉ có những gói bị ảnh hưởng. //Một ví dụ về một sự thay đổi lớn như vậy là việc chuyển đổi từ một giao diện dòng hướng đến một giao diện cửa sổ định hướng.

Nếu không có thay đổi giao diện chính được lên kế hoạch, thay đổi các dịch vụ hệ thống nên được các nguyên tắc hướng dẫn, chứ không phải là thay đổi giao diện. Các lớp ranh giới sau đó nên được đặt cùng với các lớp thực thể và kiểm soát có liên quan theo chức năng. Bằng cách này, nó sẽ được dễ dàng để xem những gì các lớp ranh giới bị ảnh hưởng nếu một thực thể hoặc kiểm soát lớpnhất định được thay đổi.

Lớp ranh giới bắt buộc không được có chức năng liên quan đến bất kỳ thực thể hoặc kiểm soát lớp, nên được đặt trong gói (packages) riêng, cùng với các lớp ranh giới thuộc về cùng một giao diện.

//Nếu một lớp ranh giới có liên quan đến một dịch vụ tùy chọn, nhóm nó trong một hệ thống phụ riêng biệt với các lớp học hợp tác để cung cấp các dịch vụ. Các hệ thống con sẽ bản đồ vào một thành phần tùy chọn đó sẽ được cung cấp khi các chức năng tùy chọn được đặt mua.

**10:**

Một gói cần được xác định cho mỗi nhóm lớp có liên quan theo chức năng. Có một số tiêu chí thực tế có thể được áp dụng khi xét xử nếu hai lớp có liên quan theo chức năng. Đây là, trong thứ tự quan trọng giảm bớt:

- Nếu những thay đổi trong một lớp 'hành vi và / hoặc cấu trúc đòi hỏi những thay đổi trong lớp khác, hai lớp có chức năng liên quan.

//- Có thể tìm hiểu xem một lớp có chức năng liên quan khác bằng cách bắt đầu với một lớp - ví dụ, một lớp thực thể - và kiểm tra tác động của nó bị xóa khỏi hệ thống. Bất kỳ lớp nào không cần thiết là kết quả của việc loại bỏ lớp được bằng cách nào đó kết nối với các lớp học đã bị gỡ. Ngoài ra, chúng tôi có nghĩa là các lớp chỉ được sử dụng bởi các lớp bị loại bỏ, hoặc là chính nó phụ thuộc vào các lớp đã bị gỡ.

- Hai đối tượng có thể có chức năng liên quan nếu chúng tương tác với một số lượng lớn các tin nhắn, hoặc có một liên lạc ở phức tạp.

- Một lớp ranh giới có thể là chức năng liên quan đến một lớp thực thể đặc biệt nếu các chức năng của các lớp ranh giới để trình bày các lớp thực thể.

- Hai lớp có thể có chức năng liên quan nếu chúng tương tác với, hoặc bị ảnh hưởng bởi những thay đổi trong các diễn viên cùng. Nếu hai lớp này không liên quan đến các tác nhân, họ cũng không nên nằm trong cùng một gói. Quy tắc cuối cùng có thể, tất nhiên, được bỏ qua vì những lý do quan trọng hơn.

//- Hai lớp có thể có được chức năng liên quan, nếu họ có mối quan hệ với thứ khác (hiệp hội, tụ, vv). Tất nhiên, tiêu chuẩn này không thể thực hiện vô thức nhưng có thể được sử dụng khi không có tiêu chí khác được áp dụng.

- Một lớp có thể là chức năng liên quan đến các lớp tạo ra trường hợp của nó.

**11:**

Trong phân tích kiến trúc, chúng tôi đã thảo luận phụ thuộc gói. Bây giờ chúng ta hãy nhìn vào phụ thuộc gói chi tiết hơn và xem như thế nào khả năng hiển thị có thể được xác định.

Tầm nhìn có thể được xác định cho các yếu tố gói theo cùng một cách nó được định nghĩa cho thuộc tính lớp và các hoạt động. Khả năng hiển thị này cho phép bạn chỉ định các gói khác có thể truy cập vào các yếu tố đó được sở hữu bởi các gói.

Tầm nhìn của một yếu tố gói có thể được thể hiện bằng cách bao gồm khả năng hiển thị một biểu tượng như một tiền tố để các tên phần tử gói.

Có ba loại khả năng hiển thị được định nghĩa trong UML:

**Public**: có thể được truy cập bên ngoài của gói sở hữu. Biểu tượng Visibility: +.

**Protected**:chỉ có thể được truy cập bởi các gói sở hữu và gói phần mềm mà kế thừa từ các gói sở hữu. Biểu tượng Visibility: #.

**Private**: chỉ có thể được truy cập bởi các lớp trong gói sở hữu. Biểu tượng Tầm nhìn: -.

Các yếu tố của công chúng về một gói tạo thành giao diện của gói. Tất cả phụ thuộc vào một gói nên phụ thuộc vào các yếu tố nào của gói.

Khả năng hiển thị cung cấp các gói hỗ trợ cho các nguyên tắc OO về đóng gói.

**12:**

Gói khớp nối là tốt và xấu: Tốt, vì khớp nối đại diện tái sử dụng, và xấu, vì đại diện cho khớp nối phụ thuộc mà làm cho hệ thống khó khăn hơn để thay đổi và phát triển. Một số nguyên tắc chung có thể được theo sau:

- Gói không phải là cross-coupled (có nghĩa là đồng phụ thuộc); Ví dụ, hai gói không nên phụ thuộc vào nhau. Trong những trường hợp này, các gói cần phải được tổ chức lại để loại bỏ các cross-phụ thuộc.

- Gói trong lớp thấp không nên phụ thuộc vào gói trong các lớp trên. Các gói chỉ nên phụ thuộc vào gói trong cùng một lớp và trong lớp thấp hơn tiếp theo. Trong những trường hợp này, các chức năng cần phải được phân hoạch lại. Một giải pháp là để nhà nước phụ thuộc về mặt giao diện, và tổ chức các giao diện trong các lớp thấp hơn.

- Nói chung, phụ thuộc không nên bỏ qua các lớp, trừ các hành vi phụ thuộc là phổ biến trên tất cả các lớp, và thay thế là chỉ cần đi qua các dẫn hoạt động trên lớp.

- Gói không nên phụ thuộc vào hệ thống con - chỉ có trên các gói khác hoặc trên giao diện.

**13:** VD????

Vài slide tiếp theo mô tả quyết định đóng gói cho hệ thống đăng ký khóa học.

Tất cả các lớp đặc biệt hỗ trợ đăng ký đã được phân chia thành các gói đăng ký.

Để đơn giản trên sơ đồ trên, chỉ có các lớp học gói đăng ký của sinh viên đã được chứng minh, và các hoạt động và các thuộc tính không được hiển thị.

**14: 15:**

**16:**

Hệ thống con có thể được sử dụng để phân vùng hệ thống vào các bộ phận có thể được độc lập:

đặt, cấu hình, hoặc giao

phát triển, miễn là giao diện không thay đổi

triển khai trên một tập hợp các nút tính toán phân phối

thay đổi mà không vi phạm các bộ phận khác của hệ thống

Hệ thống con cũng có thể được sử dụng để:

phân vùng hệ thống thành các đơn vị có thể cung cấp bảo mật hạn chế về tài chính

đại diện cho các sản phẩm hiện có hoặc các hệ thống bên ngoài trong việc thiết kế (ví dụ như các thành phần)

Hệ thống con cung cấp một "thiết kế thay thế" yếu tố: Bất kỳ hai hệ thống con (hoặc các lớp, cho rằng vấn đề) mà nhận ra các giao diện tương tự có thể hoán đổi.

Hệ thống con hỗ trợ nhiều biến thể thực hiện. Hệ thống con có thể được sử dụng khi mô hình hóa là một trong nhiều biến thể thực hiện.

Hệ thống con có thể được sử dụng để đại diện cho các thành phần từ các mẫu thực hiện trong Mô hình thiết kế.

phần mềm giao tiếp

Hỗ trợ truy cập cơ sở dữ liệu

Các loại và cấu trúc dữ liệu

tiện ích phổ biến

Sản phẩm ứng dụng cụ thể

**17:**

Ví dụ về các lớp phân tích rằng có thể phát triển thành hệ thống phụ bao gồm:

- Lớp học cung cấp các dịch vụ và / hoặc các tiện ích phức tạp. Ví dụ:

+ Động cơ tín dụng hoặc đánh giá rủi ro trong các ứng dụng tài chính

+ Động cơ đánh giá dựa trên luật lệ trong các ứng dụng thương mại

+ Dịch vụ ủy quyền bảo mật trong hầu hết các ứng dụng.

- Lớp ranh giới, cả hai cho giao diện người dùng và giao diện hệ thống bên ngoài. Nếu giao diện (s) rất đơn giản và rõ ràng, một lớp duy nhất có thể là đủ. Thông thường, tuy nhiên, các giao diện này quá phức tạp để được đại diện bằng cách sử dụng một lớp duy nhất. Họ thường đòi hỏi sự hợp tác phức tạp của nhiều lớp. Hơn nữa, các giao diện này có thể được tái sử dụng trên các ứng dụng. Kết quả là, một cách thích hợp một hệ thống phụ nhiều mô hình các giao diện này trong nhiều trường hợp.

Ví dụ về các sản phẩm hệ thống sử dụng mà bạn có thể thể hiện bằng một hệ thống phụ bao gồm:

- Phần mềm giao tiếp (middle-ware, COM / hỗ trợ CORBA)

- Hỗ trợ truy cập cơ sở dữ liệu (hỗ trợ lập bản đồ RDBMS)

- Các loại và cấu trúc dữ liệu (sách, danh sách, hàng đợi)

- Tiện ích phổ biến (các thư viện toán học)

- Sản phẩm ứng dụng cụ thể (hệ thống thanh toán, kê khai lịch trình)

//Giao diện và hệ thống con cung cấp các tách cần thiết giữa giao diện và thực hiện mô hình (trong thiết kế) những thành phần làm cho thực hiện.

**//18:**

Khi các lớp phân tích phức tạp, như vậy mà nó xuất hiện để thể hiện hành vi đó không thể là trách nhiệm của một lớp diễn xuất đơn lẻ, hoặc các trách nhiệm có thể cần phải được tái sử dụng, các lớp phân tích cần được tinh chế thành một hệ thống phụ. Đây là một quyết định chủ yếu dựa trên sự phỏng đoán được hướng dẫn bằng kinh nghiệm. Các đại diện thực tế có thể mất một vài lần lặp để ổn định.

Như đã thảo luận trước đó, việc sử dụng một hệ thống phụ cho phép giao diện được định nghĩa và ổn định, trong khi để lại các chi tiết thiết kế của giao diện để thực hiện vẫn còn tiềm ẩn trong khi định nghĩa của nó phát triển.

Quyết định làm một cái gì đó một hệ thống phụ thường được thúc đẩy bởi những kiến thức và kinh nghiệm của các kiến trúc sư. Kể từ khi nó có xu hướng để có một ảnh hưởng mạnh mẽ trên các phân vùng của không gian giải pháp, các quyết định cần phải được thực hiện trong bối cảnh của toàn bộ mô hình. Nó là kết quả của kiến thức thiết kế chi tiết hơn, cũng như việc áp đặt những hạn chế áp đặt bởi các môi trường thực hiện.

Khi một lớp phân tích được phát triển thành một hệ thống con, trách nhiệm đã được phân bổ cho các "siêu nhân" lớp phân tích sau đó được phân bổ cho các hệ thống con và một giao diện liên quan (có nghĩa là, chúng được sử dụng để xác định các hoạt động giao diện). Các chi tiết về cách hệ thống con thực sự thực hiện các trách nhiệm (có nghĩa là, các hoạt động giao diện) bị hoãn lại cho đến khi hệ thống con Design

**19:**

Hệ thống con và các gói được rất giống nhau, nhưng khác nhau ở một số cách cần thiết. Một hệ thống con cung cấp các giao diện mà các hành vi đó có thể được truy cập. Gói cung cấp không có hành vi; họ chỉ đơn giản chứa đựng những điều mà có hành vi. Gói giúp tổ chức và bộ điều khiển của lớp đó là cần thiết trong phổ biến, nhưng đó không phải là thực sự hệ thống con. Các gói được chỉ được sử dụng để tổ chức mô hình quản lý và cấu hình.

Hệ thống con hoàn toàn gói gọn nội dung của họ, cung cấp các hành vi chỉ thông qua giao diện của chúng. Phụ thuộc vào một hệ thống con là trên giao diện của nó (s), không phải về những nội dung cụ thể hệ thống phụ. Với gói, phụ thuộc vào các yếu tố cụ thể là trong các gói.

Với hệ thống con, các nội dung và hành vi bên trong của một hệ thống phụ có thể thay đổi với sự tự do hoàn toàn miễn là giao diện của hệ thống phụ vẫn không đổi. Với gói, nó là không thể thay thế các gói cho nhau, trừ khi họ có các lớp nào giống nhau. Các lớp public và các hoạt động public của nó bị đóng băng do phụ thuộc vào các lớp bên ngoài có vào chúng. Vì vậy, các nhà thiết kế khó khăn hơn để loại bỏ những lớp hoặc thay đổi hành vi của họ nếu một ý tưởng tốt hơn xuất hiện.

Lưu ý: Ngay cả khi sử dụng các gói, điều quan trọng là bạn che giấu thực hiện từ yếu tố bên ngoài để gói. Tất cả phụ thuộc vào một gói nên được vào lớp nào của gói. Lớp công chúng có thể được coi là giao diện của gói và phải được quản lý như vậy (ổn định sớm).

**20: 21: 22: 23:**

**24:**

Tái sử dụng có thể được phát hiện trong các hệ thống phần mềm hiện nay. Có thể có các yếu tố thiết kế cần thiết bởi nhiều hơn một hệ thống phụ hoặc gói. Slide trên là có nghĩa là để trực quan đại diện những gì sẽ xảy ra khi sử dụng lại được phát hiện:

Xây dựng các ứng dụng đầu tiên và một số bộ phận chung.

Xây dựng các ứng dụng thứ hai, và bạn thấy rằng một số bộ phận của đầu tiên có thể được tái sử dụng, nhưng các bộ phận không được thiết kế để được sử dụng lại. (Một số được sử dụng lại bất kỳ cách nào, với sự hỗn loạn như kết quả). Phải thừa nhận rằng một cái gì đó đang được sử dụng bởi một yếu tố mà các yếu tố khác có thể tìm thấy hữu ích (sơ đồ đầu tiên).

Lấy thí sinh yếu tố thiết kế tái sử dụng (các lớp học, các gói, hoặc hệ thống con) và làm cho họ tái sử dụng. Điều này có thể liên quan đến việc thay đổi tên của họ, làm cho họ tổng quát hơn, cải thiện tài liệu của họ, di chuyển chúng đến một lớp chức năng phổ biến, vv Về cơ bản, các ứng cử viên tái sử dụng thực thể được "đẩy xuống" để một lớp thấp hơn trong kiến trúc nên các yếu tố khác có thể truy cập nó . Ban đầu, chỉ có các khách hàng ban đầu của nguyên tố này có thể sử dụng nó (sơ đồ thứ hai).

Bây giờ bạn có thể sử dụng chúng cho các ứng dụng mới.

Khi các ứng dụng đầu tiên được nâng cấp, các phiên bản cũ của các phần tử có thể được thay thế bằng những người tái sử dụng (sơ đồ thứ ba).

Điều quan trọng là các kiến trúc sư để nhìn vào bên trong và giữa các yếu tố thiết kế cho hành vi phổ biến (ví dụ, hợp tác thông thường), kéo nó ra nếu có thể. Đây là một hoạt động nhặt rác tái sử dụng mà thuộc Xác định yếu tố thiết kế chiếc ô.

Lưu ý: Để xem một ví dụ về việc tái sử dụng một cơ chế an ninh, tham khảo các tab Cơ chế bảo mật trong các thông tin khác Phụ lục.

**25:**

Bảng trên là một ví dụ về cách bạn có thể ghi lại các bản đồ của các lớp phân tích để thiết kế các yếu tố. Điều này có thể được tinh chế là quá trình kế tiếp.

**26:**

**27:**

Đây là một sự lặp lại của các slide đầu tiên được giới thiệu trong phân tích kiến trúc. Nó được bao gồm ở đây như là một xem xét.

Trong phân tích kiến trúc, việc tập trung vào các lớp cấp trên (có nghĩa là, các ứng dụng và các lớp kinh doanh cụ thể). Trong Xác định yếu tố thiết kế, tập trung vào các lớp dưới.

Các nguyên tắc lớp mô tả ban đầu đối với các gói cũng áp dụng cho các hệ thống con.

Specific functionality: chức năng cụ thể

Specific funcionality: chức năng chung

1. Hệ thống con ứng dụng riêng biệt mà tạo nên một ứng dụng - có chứa các phần mềm giá trị gia tăng được phát triển bởi tổ chức.
2. Kinh doanh cụ thể - có chứa một số hệ thống con tái sử dụng cụ thể cho các loại hình kinh doanh.
3. Middleware - cung cấp các hệ thống con cho các lớp học và các dịch vụ tiện ích độc lập nền tảng cho tính toán phân tán đối tượng trong môi trường không đồng nhất,....
4. Phần mềm hệ thống - có chứa các phần mềm cho các cơ sở hạ tầng thực tế như hệ thống điều hành, giao diện phần cứng cụ thể, trình điều khiển thiết bị, ....

**28:**

**29:**

Các gói đăng ký phụ thuộc vào các gói Artifacts Đại học cho các khái niệm trừu tượng cốt lõi, các gói giao diện hệ thống bên ngoài cho các giao diện hệ thống bên ngoài, và các khung giao diện và các gói giao diện bảo mật cho các khuôn khổ an ninh.

Chú ý cách phụ thuộc gói phù hợp với sự phụ thuộc lớp.