


Nhập môn Công Nghệ Phần Mềm
Mô hình hoá Phần mềm với UML

Nội dung

- ♦ Giới thiệu về UML
- ♦ Use-case diagrams
- ♦ Activity diagrams
- ♦ Interaction diagrams
- ♦ Class diagrams




2

IBM

UML là gì?

- ♦ UML là một ngôn ngữ để
 - Trực quan hoá (Visualizing)
 - Xác định (Specifying)
 - Xây dựng (Constructing)
 - Tài liệu hoá (Documenting)

các nhân tố của một hệ thống phần mềm.



3

IBM

UML là một ngôn ngữ cho trực quan hoá

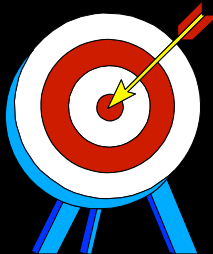
- ♦ Giao tiếp qua các mô hình khái niệm dễ phát sinh lỗi trừ khi tất cả mọi người đều sử dụng chung ngôn ngữ
- ♦ Có những mặt về phần mềm bạn không thể hiểu được trừ khi bạn xây dựng các mô hình.
- ♦ Một mô hình rõ ràng sẽ giúp cho việc giao tiếp tốt hơn.



IBM

UML là một ngôn ngữ cho việc xác định

- ♦ UML xây dựng các mô hình chính xác, rõ ràng và đầy đủ



IBM

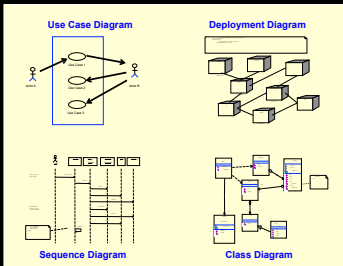
UML là một ngôn ngữ để xây dựng

- ♦ Các mô hình UML có thể được kết nối trực tiếp đến các ngôn ngữ lập trình khác nhau.
 - Map đến Java, C++, Visual Basic, ...
 - Các bảng trong một RDBMS hoặc lưu trữ trong một OODBMS
 - Permits forward engineering
 - Permits reverse engineering

IBM

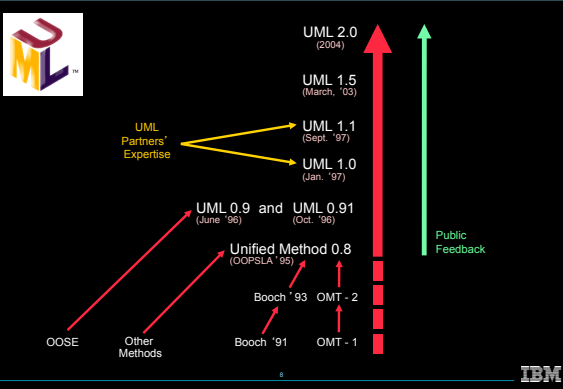
UML là một ngôn ngữ để lập tài liệu

- UML hỗ trợ tài liệu hoá về kiến trúc hệ thống, các yêu cầu, kiểm thử, kế hoạch dự án, và quản lý phát hành.



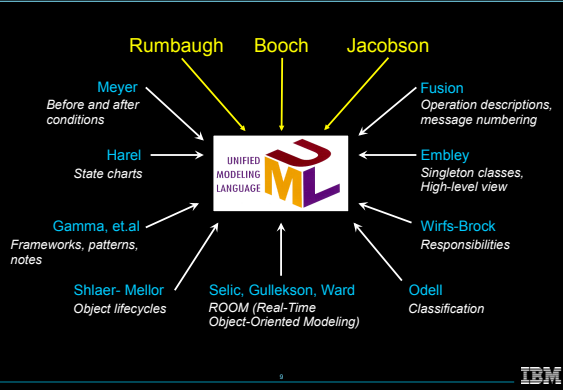
IBM

Lịch sử của UML



IBM


Đầu vào của UML



IBM

Nội dung

- ♦ Giới thiệu về UML
- ♦ Use-case diagrams
- ♦ Activity diagrams
- ♦ Interaction diagrams
- ♦ Class diagrams

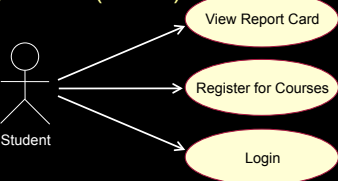


10

IBM

Thế nào là một mô hình trường hợp sử dụng?

- ♦ Một mô hình mô tả các yêu cầu chứng năng của hệ thống dưới dạng các trường hợp sử dụng.
- ♦ Một mô hình của các chức năng chính (use-cases) của hệ thống và các môi trường của nó (actors).

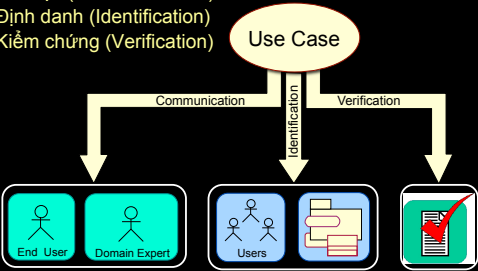


11

IBM

Lợi ích của mô hình trường hợp sử dụng là gì?

- ♦ Liên lạc (Communication)
- ♦ Định danh (Identification)
- ♦ Kiểm chứng (Verification)




12


IBM

Các khái niệm chính trong mô hình hoá trường hợp sử dụng

- ♦ Một tác nhân biểu diễn bất cứ đối tượng nào tương tác với hệ thống.
- ♦ Một trường hợp sử dụng mô tả một chuỗi các sự kiện, được thực hiện bởi hệ thống, mà mang lại một kết quả quan sát được có giá trị với một tác nhân nào đó.



Actor



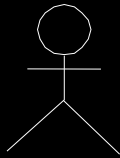
Use Case

13

IBM

Tác nhân là gì?

- ♦ Các tác nhân đại diện cho vai trò mà một người dùng hệ thống có thể làm.
- ♦ Có thể là con người, máy móc, hay một hệ thống khác.
- ♦ Có thể chủ động trao đổi thông tin với hệ thống.
- ♦ Có thể là nguồn thông tin.
- ♦ Có thể là một người nhận thông tin thụ động.
- ♦ Tác nhân không phải là một phần của hệ thống.
 - Tác nhân là yếu tố bên ngoài.




Actor

14

IBM

Trường hợp sử dụng là gì?

- ♦ Định nghĩa một tập các trường hợp sử dụng, mỗi trường hợp là một chuỗi các hành động mà hệ thống thực hiện để thu được một kết quả có ý nghĩa với một tác nhân nào đó.
 - Trường hợp sử dụng mô hình hoá sự giao tiếp giữa một hoặc nhiều tác nhân với hệ thống
 - Trường hợp sử dụng mô tả các hành động mà hệ thống cần thực hiện để cung cấp một cái gì đó có ý nghĩa với tác nhân



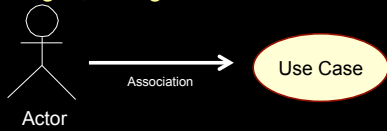
Use Case

15

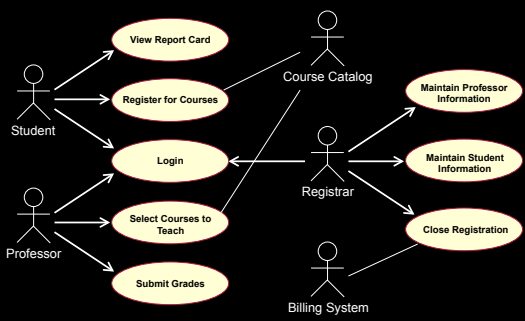
IBM

Các trường hợp sử dụng và tác nhân

- ♦ Một trường hợp sử dụng mô hình hoá một giao tiếp giữa tác nhân và hệ thống.
- ♦ Một trường hợp sử dụng được kích hoạt bởi một tác nhân để gọi một chức năng nào đó trong hệ thống.



Bạn hiểu sơ đồ sau như thế nào?



Mối quan hệ trong các trường hợp sử dụng

- ♦ Giữa tác nhân và trường hợp sử dụng
 - Tác nhân sử dụng
- ♦ Tổng quát hoá các tác nhân
 - Các dạng của người dùng
- ♦ Khuôn mẫu trường hợp sử dụng
 - <<extend>>
 - Optional
 - <<include>>
 - Mandatory
- ♦ Khuôn mẫu là một cơ chế mở rộng UML để chỉ một loại hành vi

Các biến thể trường hợp sử dụng: include and extend

- ♦ *include* được sử dụng khi bạn có một phần hành vi nào đó tương tự nhau trong nhiều hơn một trường hợp sử dụng
 - Sử dụng trong một hay nhiều Use Cases khác nhau để tránh sự lặp lại
 - Là một phần quan trọng của một use case
 - <<include>>
- ♦ *Extend* sử dụng khi bạn có một Use Case thêm chức năng vào một Use Case khác
 - Bất cứ Use Case nào có thể có nhiều hơn một extend
 - Sử dụng khi mô tả sự biến đổi hoặc bổ sung cho một hành vi bình thường

19

IBM

Ví dụ về các biến thể Use Case

```
graph TD; OA([Open account]) -.->|<<include>>| SCDS([Supply Customer Data]); PO([Place order]) -.->|<<include>>| SCDS; PO -.->|<<extend>>| POB([Place back order]); POB -.->|<<include>>| AD([Arrange delivery]); SCDS -- shared functionality --> POB; POB -- additional functionality --> POB
```

20

IBM

Bài tập 1 – Use Case Diagram

- ♦ Draw a use case diagram for the space invader game below:

<http://www.neave.com/games/invaders/>

21

IBM

Exercise 1 Solution

A UML Use Case Diagram showing a stick figure actor labeled 'Player' connected to five use case ovals. The use cases are: 'Move Left', 'Move Right', 'Fire Laser', 'View High Scores', and 'Play Game'. The connections are straight lines from the actor to each use case.

IBM

Exercise 2

Consider Sonic the hedgehog.

1. What can sonic do?
2. What are the use cases?
3. Are there any relationships
4. Draw the use case diagram

A screenshot from the Sonic the Hedgehog game showing Sonic running on a green hill with a blue sky and clouds in the background. The game interface shows a score of 0, a time of 19:47, and a level indicator.

1. Move left
2. Move right
3. Jump
4. Jump left
5. Jump right
6. Spin Dash
7. Pause

<http://www.ebaumsworld.com/games/play/1108/>

IBM

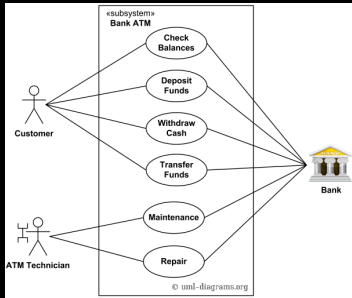
Exercise 2 – Possible Solution

A UML Use Case Diagram for 'Sonic the hedgehog'. A stick figure actor labeled 'Player' is connected to several use case ovals: 'Move Left', 'Move Right', 'Jump', 'Spin Dash', and 'Pause Game'. There are also use case ovals for 'Jump Left' and 'Jump Right'. 'Move Right' is connected to 'Jump Left' with a relationship labeled '<<extend>>'. 'Jump' is connected to 'Jump Right' with a relationship labeled '<<extend>>'. A note box labeled 'What about: New game, Choose character, etc' is connected to the 'Player' actor.

IBM

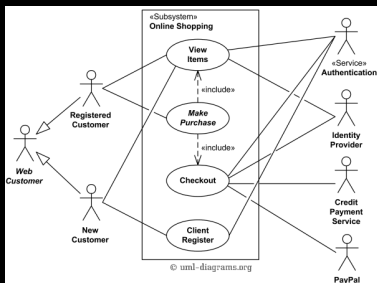
Bài tập 3: Use-case Diagram for Bank ATM

- Customer uses a bank ATM to check balances of his/her bank accounts, deposit funds, withdraw cash and/or transfer funds (use cases). ATM Technician provides maintenance and repairs to the ATM.

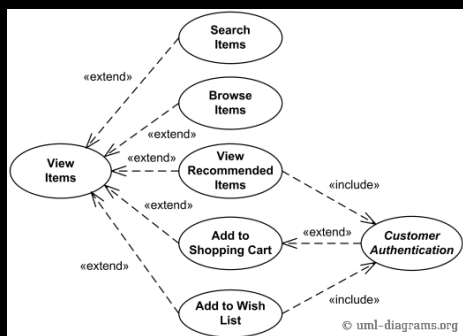


Bài tập 4: Use-case Diagram for Online Shopping

- Web customer actor uses some web site to make purchases online. Top level use cases are View Items, Make Purchase and Client Register.




Bài tập 4: Use-case Diagram for Online Shopping



Nội dung

- ♦ Giới thiệu về UML
- ♦ Use-case diagrams
- ♦ Activity diagrams
- ♦ Interaction diagrams
- ♦ Class diagrams



28IBM

Một biểu đồ hoạt động là gì?

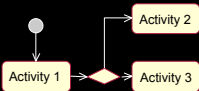
- ♦ Một biểu đồ hoạt động trong một mô hình trường hợp sử dụng có thể được sử dụng để nắm bắt các hoạt động và hành động được thực hiện trong một trường hợp sử dụng.
- ♦ Nó thực chất là một biểu đồ dòng chảy, chỉ ra dòng chảy điều khiển từ một hoạt động/hành động này sang hoạt động/hành động khác.

Flow of Events

This use case starts when the Registrar requests that the system close registration.

1. The system checks to see if registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress.

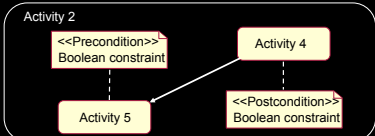
2. For each course offering, the system checks if a professor has signed up to teach the course offering and at least three students have registered. If so, the system commits the course offering for each schedule that contains it.



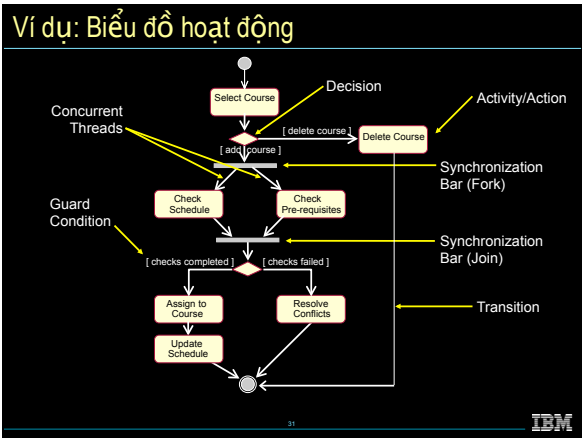
29IBM

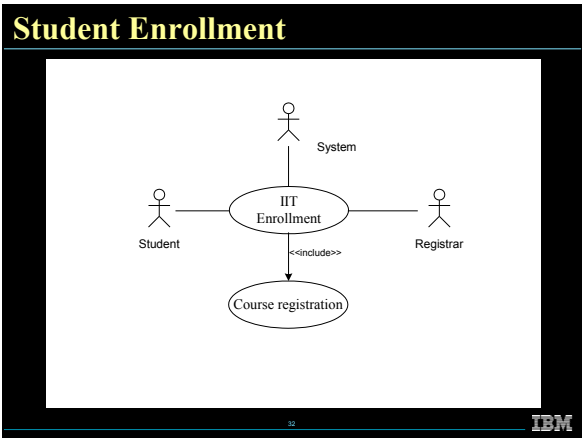
Một hoạt động là gì?

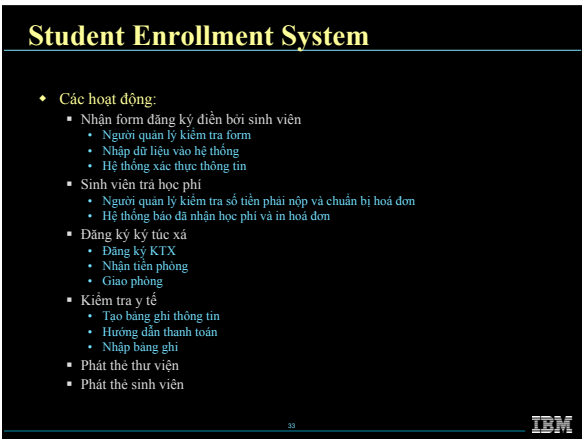
- ♦ Một đặc tả hành vi được thể hiện như một luồng thực hiện theo trình tự các đơn vị liên quan.
 - Các đơn vị liên quan bao gồm các hoạt động lồng nhau và các hành động đơn lẻ.
- ♦ Có thể chứa các ràng buộc biểu thức boolean khi hoạt động được kích hoạt hay kết thúc

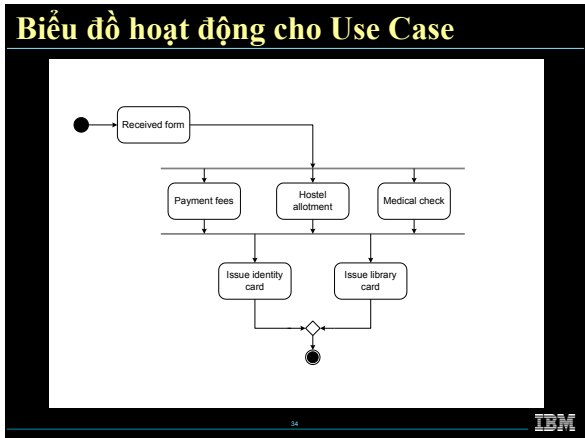


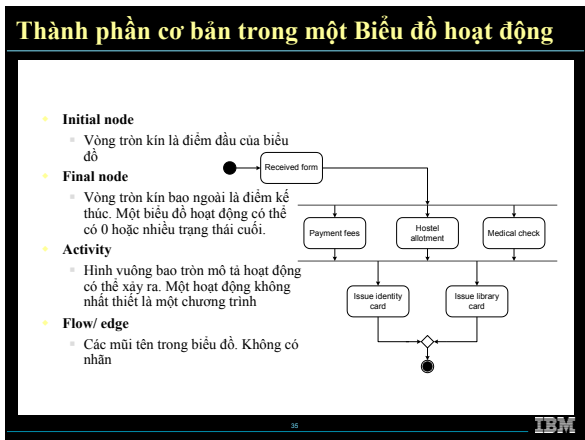
30IBM

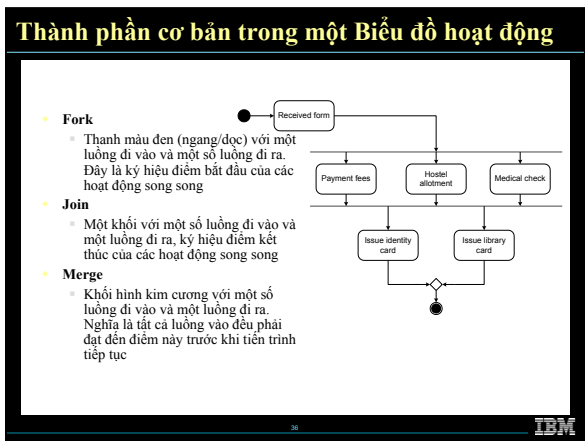












Thành phần cơ bản trong một Biểu đồ hoạt động

- ♦ Khác nhau giữa Join và Merge
 - join khác với merge ở chỗ là join đồng bộ hoá hai luồng vào và tạo 1 luồng ra đơn. Luồng ra từ join không thể chạy cho đến khi tất cả luồng vào đều kết thúc.
 - merge chuyển bất kỳ luồng nào qua thẳng nó. Nếu hai hay nhiều luồng vào được nhận bởi một merge, hành động được trỏ đến bởi luồng ra của nó được thực hiện hai hay nhiều lần

05 October, 2007

Information System Design IT60105, Autumn 2007

37

IBM

Thành phần cơ bản trong một Biểu đồ hoạt động

- ♦ Decision
 - Hình kim cương với một luồng vào và vài luồng ra. Luồng ra bao gồm điều kiện trạng thái yes/no
- ♦ Flow final
 - Hình tròn với dấu X. Thành phần này chỉ ra điểm kết thúc của tiến trình
- ♦ Swim lane
 - Một phân vùng trong biểu đồ hoạt động thể hiện bởi đường đứt, gọi là đường bơi. Đường bơi này có thể dọc hoặc ngang

38

IBM

Biểu đồ hoạt động chi tiết

```
graph TD; Start(( )) --> R1((1)); R1 --> P[Payment fees]; R1 --> H[Hostel allotment]; R1 --> M[Medical check]; P --> J1(( )); H --> J1; M --> J1; J1 --> I1[Issue identity card]; J1 --> I2[Issue library card]; I1 --> F2((2)); I2 --> F2; F2 --> End((( )))
```

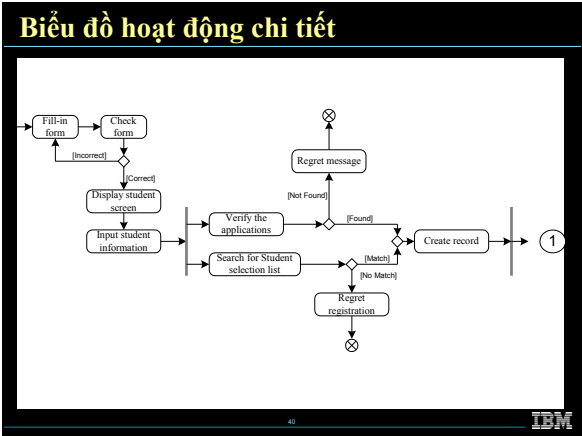
39

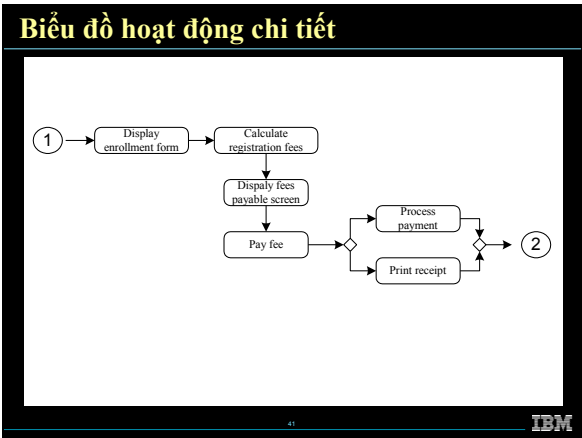
IBM

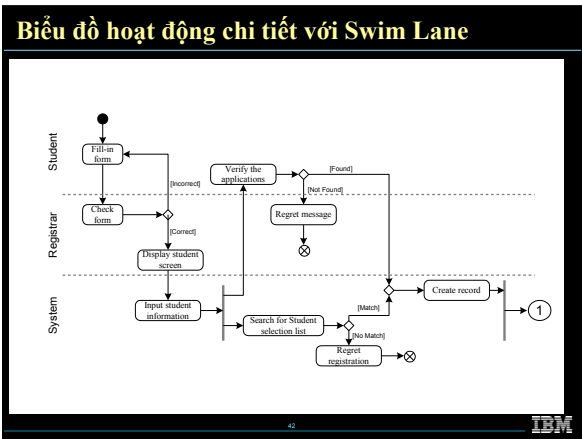
Page 13

Module 2 - Principles of Visual Modeling

13







Bài tập 3: Use-case Diagram for Process Shopping Order

- Requested order is input parameter of the activity. After order is accepted and all required information is filled in, payment is accepted and order is shipped.

43

IBM

Bài tập 3: Activity Diagram for Process Shopping Order

- Requested order is input parameter of the activity. After order is accepted and all required information is filled in, payment is accepted and order is shipped.

```
graph TD
    Start(( )) --> ReceiveOrder[Receive Order]
    RequestedOrder[Requested Order] --> ReceiveOrder
    ReceiveOrder -- "[order accepted]" --> FillOrder[Fill Order]
    ReceiveOrder -- "[order rejected]" --> End(( ))
    FillOrder --> Fork(( ))
    Fork --> SendInvoice[Send Invoice]
    Fork --> ShipOrder[Ship Order]
    SendInvoice --> Invoice[Invoice]
    Invoice --> AcceptPayment[Accept Payment]
    ShipOrder --> Join(( ))
    AcceptPayment --> Join
    Join --> CloseOrder[Close Order]
    CloseOrder --> End2(( ))
```

44

IBM

Bài tập 4: Activity Diagram for Online Shopping

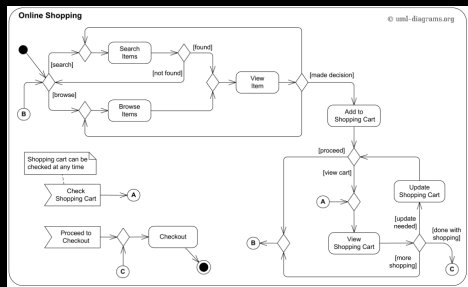
- Online customer can browse or search items, view specific item, add it to shopping cart, view and update shopping cart, do checkout. User can view shopping cart at any time.

45

IBM

Bài tập 4: Activity Diagram for Online Shopping

- Online customer can browse or search items, view specific item, add it to shopping cart, view and update shopping cart, do checkout. User can view shopping cart at any time.



Nội dung

- Giới thiệu về UML
- Use-case diagrams
- Activity diagrams
- Interaction diagrams**
- Class diagrams



Đối tượng cần hợp tác

- Đối tượng không có giá trị nếu chúng không hợp tác để giải quyết vấn đề.
 - Mỗi đối tượng chịu trách nhiệm cho trạng thái và hành vi của nó.
 - Không có đối tượng nào có thể thực hiện mọi trách nhiệm riêng của mình.
- Các đối tượng tương tác với nhau thế nào?
 - Tương tác qua các messages.

Đối tượng tương tác qua Messages

- ♦ Thông điệp cho thấy cách một đối tượng yêu cầu đối tượng khác thực hiện một số hoạt động.

```
graph LR; RC[RegistrationController] -- "getCourseOfferings(forSemester)" --> CCS[CourseCatalogSystem];
```

49

IBM

Biểu đồ tương tác là gì?

- ♦ Một dạng biểu đồ chung được áp dụng để nhấn mạnh sự tương tác của đối tượng
 - Biểu đồ tuần tự (Sequence Diagram)
 - Biểu đồ giao tiếp (Communication Diagram)
- ♦ Các biến thể
 - Biểu đồ thời gian (Timing Diagram)
 - Biểu đồ tổng quan tương tác (Interaction Overview Diagram)

50

IBM

Biểu đồ tương tác

- ♦ **Biểu đồ tuần tự**
 - View hướng thời gian của tương tác đối tượng
- ♦ **Biểu đồ giao tiếp**
 - View cấu trúc của đối tượng thông điệp

Sequence Diagrams


Communication Diagrams

51

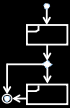
IBM

Biểu đồ tương tác

- ♦ **Biểu đồ thời gian**
 - View ràng buộc thời gian của các messages liên quan trong tương tác
- ♦ **Biểu đồ tổng quan tương tác**
 - View ở mức cao của các bộ tương tác kết hợp thành chuỗi logic.



Timing Diagrams



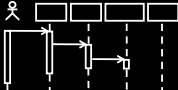
Interaction Overview Diagrams

52

IBM

Biểu đồ tuần tự là gì?

- ♦ **Biểu đồ tuần tự** là một biểu đồ tương tác nhấn mạnh đến thứ tự theo thời gian của messages.
- ♦ **Biểu đồ thể hiện:**
 - Các đối tượng tham gia vào tương tác.
 - Tuần tự của các messages trao đổi.



Sequence Diagram

53

IBM

Ví dụ: Biểu đồ tuần tự

```
sequenceDiagram
    actor Student
    participant RegisterForCoursesForm
    participant RegistrationController
    participant CourseCatalogSystem
    participant CourseCatalog

    Student->>RegisterForCoursesForm: 1: create schedule()
    activate RegisterForCoursesForm
    RegisterForCoursesForm->>RegistrationController: 2: get course offerings()
    activate RegistrationController
    RegistrationController->>CourseCatalogSystem: 3: get course offerings(forSemester)
    activate CourseCatalogSystem
    CourseCatalogSystem->>CourseCatalog: 4: get course offerings()
    activate CourseCatalog
    CourseCatalog-->>CourseCatalogSystem: 
    deactivate CourseCatalog
    CourseCatalogSystem-->>RegistrationController: 
    deactivate CourseCatalogSystem
    RegistrationController-->>RegisterForCoursesForm: 5: display course offerings()
    deactivate RegistrationController
    RegisterForCoursesForm-->>Student: 6: display blank schedule()
    deactivate RegisterForCoursesForm
```

ref

Select Offerings

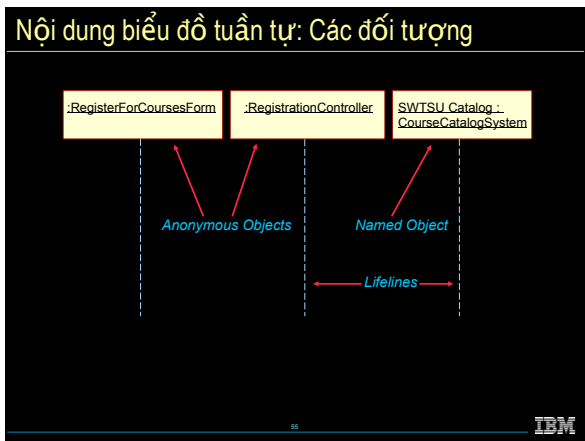
54

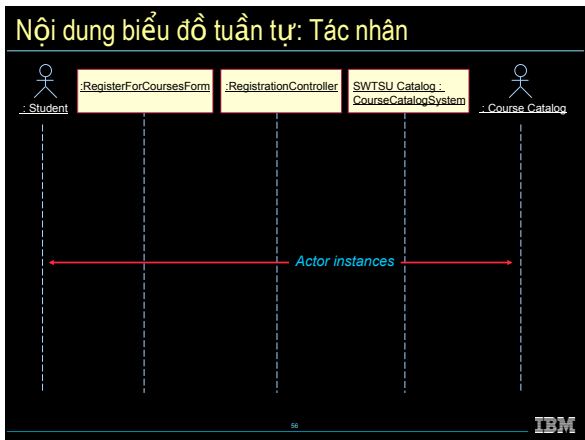
IBM

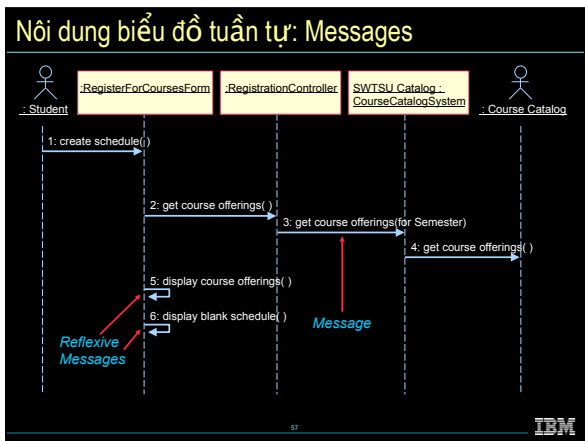
Page 18

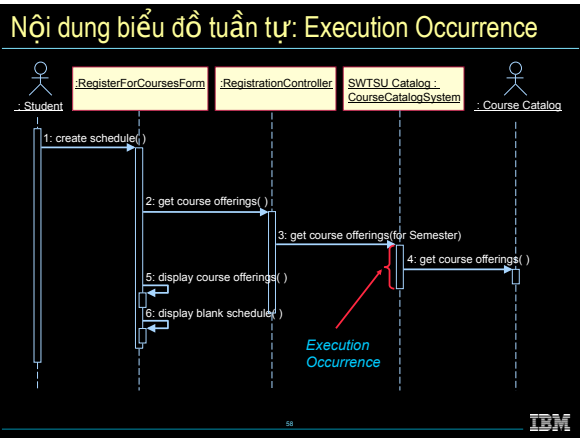
Module 2 - Principles of Visual Modeling

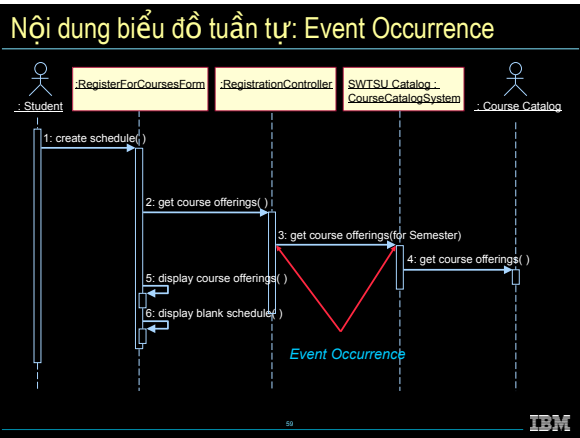
18

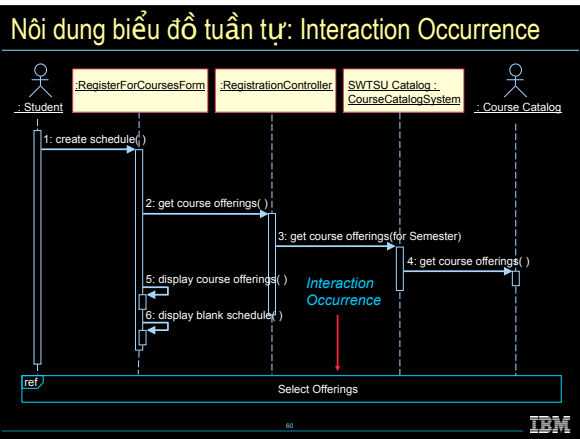












Nội dung

- ♦ Giới thiệu về UML
- ♦ Use-case diagrams
- ♦ Activity diagrams
- ♦ Interaction diagrams
- ♦ Class diagrams

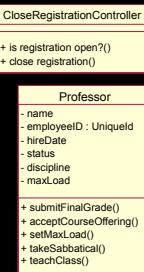
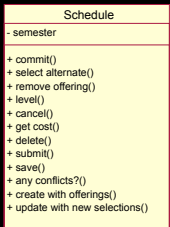
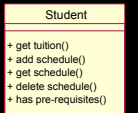
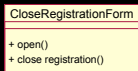


61



Biểu đồ lớp là gì

- ♦ Là view tĩnh của một hệ thống



62

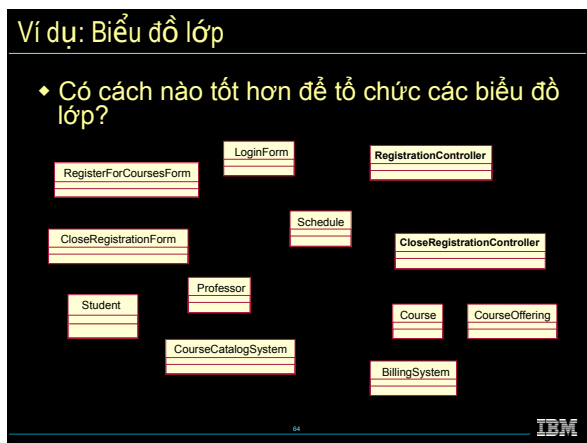


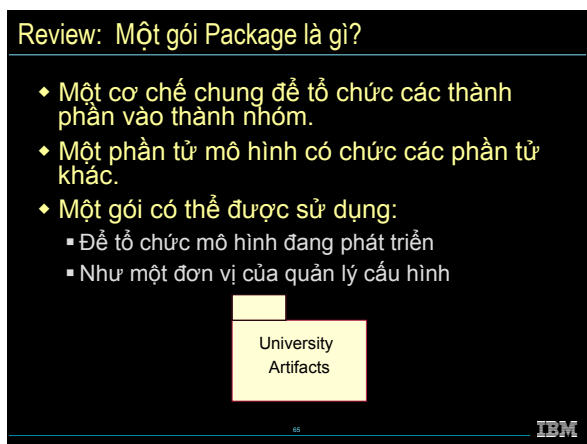
Sử dụng Biểu đồ lớp

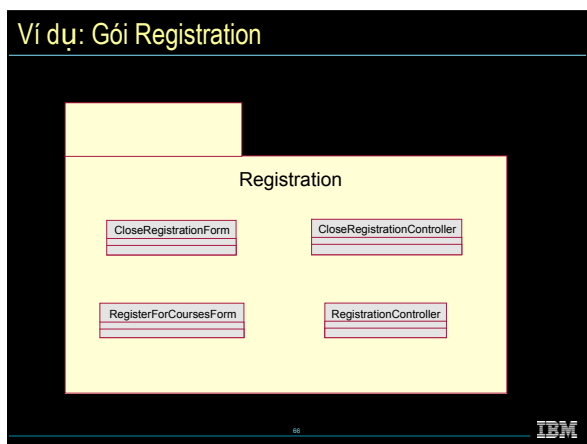
- ♦ Khi mô hình hoá view tĩnh của một hệ thống, các biểu đồ lớp được sử dụng theo một trong ba cách, để mô hình:
 - Các thành phần của một hệ thống (The vocabulary of a system)
 - Sự hợp tác (Collaborations)
 - Lược đồ cơ sở dữ liệu logic (A logical database schema)

63










Nội dung

- ♦ Các biểu đồ lớp
- ♦ Quan hệ lớp
 - ★
 - Kết hợp
 - Kết tập
 - Kế thừa

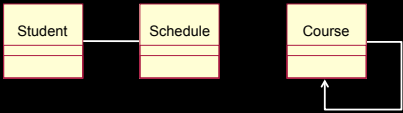


67

IBM

Kết hợp (Association) là gì?

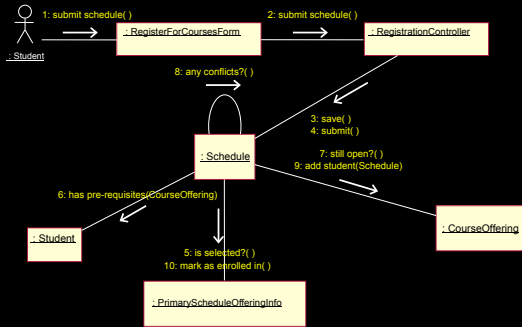
- ♦ Các mối quan hệ ngữ nghĩa giữa hai hay nhiều phân loại xác định các kết nối giữa các instances.
- ♦ Một mối quan hệ cấu trúc xác định rằng đối tượng của một lớp được kết nối với các đối tượng của lớp khác



68

IBM

Ví dụ: What Associations Can You Find?



69

IBM

Page 23

Module 2 - Principles of Visual Modeling

23

Bội số (Multiplicity) là gì?

- ♦ Bội số là số các instances của một lớp liên quan đến một instance của lớp khác.
- ♦ Với mỗi quan hệ kết hợp, có hai bội số cần quyết định ở mỗi đầu của quan hệ.
 - Với mỗi instance của Professor, nhiều Course Offerings có thể được dạy.
 - Với mỗi instance của Course Offering, có thể có một hoặc không có Professor nào.



70



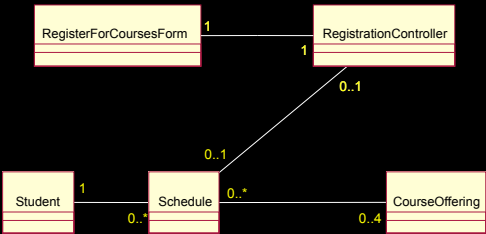
Các giá trị bội số

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

71



Ví dụ: Bội số




72



Nội dung


- ♦ Biểu đồ lớp
- ♦ Các quan hệ của lớp
 - Kết hợp
- ★
 - Kết tập
 - Kế thừa



73IBM

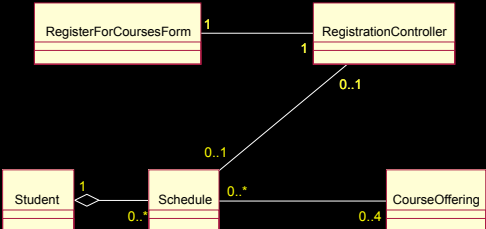
Quan hệ Kết tập (Aggregation) là gì?

- ♦ Một dạng đặc biệt của quan hệ kết hợp mô hình một mối quan hệ toàn phần giữa tập hợp (toàn bộ) và các phần của nó.
 - Kết tập là mối quan hệ “là một phần của”.
- ♦ Bội số được biểu diễn như trong quan hệ kết hợp.



74IBM

Ví dụ: Quan hệ kết tập



75IBM


Page 25

Module 2 - Principles of Visual Modeling

25

Nội dung

- ♦ Biểu đồ lớp
- ♦ Các quan hệ lớp
 - Kết hợp
 - Kết tập
- ★
 - Kế thừa



76IBM

Review: Kế thừa là gì?

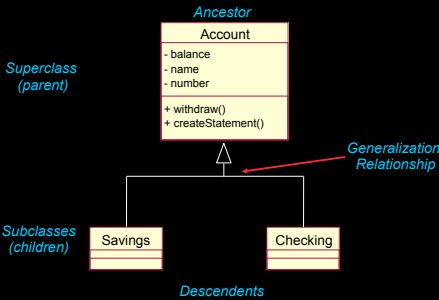
- ♦ Một quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hay nhiều lớp khác.
- ♦ Định nghĩa sự phân cấp về trừu tượng hoá trong đó một lớp con kế thừa từ một hay nhiều lớp cha.
 - Đơn kế thừa
 - Đa kế thừa
- ♦ Là mối quan hệ “là một dạng của”.

77

IBM

Ví dụ: Đơn Kế thừa

- ♦ Một lớp kế thừa từ một lớp khác.



78IBM

Ví dụ: Đa kế thừa

♦ Một lớp có thể kế thừa từ một số lớp khác.

```
classDiagram
    FlyingThing <|-- Airplane
    FlyingThing <|-- Helicopter
    FlyingThing <|-- Bird
    Animal <|-- Bird
    Animal <|-- Wolf
    Animal <|-- Horse
```

Sử dụng đa kế thừa chỉ khi thực sự cần thiết và luôn chú ý khi sử dụng

79 IBM
