|  |  |  |
| --- | --- | --- |
| Macintosh HD:Users:hoangnguyen:Documents:TDC:fitlog_blue.png | TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC  **Khoa Công Nghệ Thông Tin** | Macintosh HD:Users:hoangnguyen:Documents:TDC:logoTDC_blue.png |
|  |  |  |

**CHUYÊN ĐỀ PHÁT TRIỂN WEB 2** | HKI – 2019

**BÁO CÁO UML**

**Sinh viên thực hiện:**

1. Nguyễn Duy Cường
2. Nguyễn Thế Bảo
3. Phạm Xuân Phú
4. Phạm Văn Hiếu

BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| STT | Tên | Công việc | Dự kiến hoàn thành | tiến độ (%) | khó khăn gặp phải | Thuận lợi |
| 1 | Nguyễn Duy Cường | Tổng quan UML,  Lịch sử UML,  Mô hình và các hướng nhìn | 29/9/2019 | 29/09/2019  100% | -tài liệu trên internet từ  nhiều nguồn gây rối rắm thông tin | -anh em hoạt động tích cực,trao đổi liên tục qua facebook và gmail |
| 2 | Nguyễn Thế Bảo | các biểu đồ chính,  thành phần cơ bản | 29/9/2019 | -28/09/2019  50%  -29/09/2019  100% | -Các thành viên có thời gian khác nhau nên khó trao đổi. | -Các thành viên có thể trao đổi qua online  - các thành viên nhiệt tình, giúp đỡ lẫn nhau.  -Tài liệu có thể tìm thấy dễ dàng trên internet. |
| 3 | Phạm Xuân Phú | giới thiệu các chương trình vẽ UML | 29/9/2019 | -28/09/2019  50%  -29/09/2019  100% | -Có ít thời gian tìm hiểu thông tin  - Nhiều nguồn thông tin khác nhau khó tìm hiểu | -Các thành viên tích cực thực hiện công việc được phân công của mình  - Có thể trao đổi các vấn đề với nhau qua internet  - Tài liệu dễ dàng tìm thấy trên internet |
| 4 | Phạm Văn Hiếu | Cách vẽ lược đồ | 29/9/2019 | -28/09/2019  50%  -29/09/2019  100% | -Có ít thời gian tìm hiểu thông tin  - Nhiều nguồn thông tin khác nhau khó tìm hiểu | -Các thành viên tích cực thực hiện công việc được phân công của mình  - Có thể trao đổi các vấn đề với nhau qua internet  - Tài liệu dễ dàng tìm thấy trên internet |

**Mục Lục**

**Tổng quan về UM**L

1. Lịch sử UML……………………………………………………5

2. UML là gì? ……………………………………………………..5

3. Tại sao nên sử dụng UML? …………………………………….6

4. Mô hình và hướng nhìn…………………………………………7

**Biểu đồ UML**

1. Biểu đồ Use Case (Use Case Diagram)………………………..11
2. Biểu đồ hoạt động (Activity Diagram)………………………..15
3. Biểu đồ tương tác (Interaction Diagrams)…………………….18
4. Biểu đồ trạng thái (Statechart Diagram)………………………20
5. Biểu đồ lớp (Class Diagram)………………………………….21
6. Biểu đồ đối tượng (Object Diagram)………………………….22
7. Biểu đồ thành phần (Component Diagram)…………………..23
8. Biểu đồ triển khai (Deployment Diagram)……………………24

**Chương trình vẽ UML**

1. Giới thiệu về công cụ vẽ UML………………………………..24

2. Các chức năng của StarUML……………………………...25

3. Sử dụng StarUML…………………………………………..25

***Cách vẽ biểu đồ UML***

1.Biểu đồ Use-case ………………………………………………28

2. Biểu đồ lớp…………………………………………………….31

**Mục lục hình ảnh**

Hình 1…………………………………………………………………..11

Hình 2…………………………………………………………………..12

Hình 3…………………………………………………………………..13

Hình 4…………………………………………………………………..13

Hình 5…………………………………………………………………..13

Hình 6…………………………………………………………………..14

Hình 7…………………………………………………………………..14

Hình 8…………………………………………………………………..14

Hình 9…………………………………………………………………..15

Hình 10…………………………………………………………………15

Hình 11…………………………………………………………………15

Hình 12…………………………………………………………………20

Hình 13…………………………………………………………………21

Hình 14…………………………………………………………………23

Hình 15…………………………………………………………………24

Hình 16…………………………………………………………………27

Hình 17…………………………………………………………………28

**Bảng tra cứu từ viết tắt**

|  |  |  |  |
| --- | --- | --- | --- |
| Từ viết tắt | | Giải thích | trang |
| UML | | Unified Modeling Language | 5 |
| OMG | | Object Management Group | 25 |
|  | |  |  |
|  |
|  | |  |  |

**Tổng quan về UM**L

**1. Lịch sử UML**

* Những năm đầu của thập kỷ 90 có rất nhiều phương pháp phân tích, thiết kế hệ thống hướng đối tượng và cùng với chúng là các ký hiệu riêng cho từng phương pháp
* Thời kỳ này còn được biết đến với tên gọi là cuộc chiến giữa các phương pháp
* Khoảng đầu năm 94, Booch đã cải tiến phương pháp của mình trong đó có ứng dụng những ưu điểm của các phương pháp của Rumbaugh và Jacobson
* Các phương pháp đã bắt đầu hợp nhất, nhưng các kí hiệu sử dụng ở các phương pháp vẫn còn nhiều điểm khác biệt
* UML - một ngôn ngữ mô hình hóa hợp nhất đã trở thành một quy trình chuẩn của tổ chức OMG (Object Management Group)

**2. UML là gì?**

* Nó là một ngôn ngữ mô hình hóa, không phải là một phương pháp
* Một phương pháp thông thường bao gồm một tập các quy tắc, một ngôn ngữ mô hình hóa và một quy trình mô tả những công việc cần làm
* Một ngôn ngữ mô hình hóa bao gồm các ký hiệu và các cách thức sử dụng cho việc thiết kế
* Quy trình là những sự hướng dẫn các bước trong quá trình thiết kế
* Ngôn ngữ mô hình hóa là thành phần quan trọng nhất trong một phương pháp, là yếu tố kết nối giữa các thành phần

**3. Tại sao nên sử dụng UML?**

* Giúp cho việc phân tích và thiết kế
* Là công cụ giao tiếp (Communication)
* Tài liệu hóa (Documentation)

Như được đề cập trong *The Unified Modeling Language User Guide, n*gôn ngữ UML dùng để:

* *Trực quan hóa (Visualizing)*

Hiểu và giải quyết các vấn đề dễ dàng hơn

Khi trở thành một chuẩn trong việc lập mô hình, mỗi kí hiệu mang một ý nghĩa rõ ràng và duy nhất=> một nhà phát triển có thể đọc được mô hình xây dựng bằng UML do một người khác viết.

Một mô hình rõ ràng, sáng sủa làm tăng khả năng giao tiếp, trao đổi giữa các nhà phát triển

* *Đặc tả (Specifying)*

Xây dựng các mô hình một các tỉ mỉ, rõ ràng, đầy đủ ở các mức độ chi tiết khác nhau

Tất cả các công đoạn từ phân tích, thiết kế cho đến triển khai đều có các biểu đồ UML biểu diễn.

* *Xây dựng (Constructing)*

UML là một ngôn ngữ chuẩn với tập quy tắc về cú pháp riêng.

Chúng ta có thể xây dựng các công cụ để chuyển đổi các mô hình

Các mô hình của UML có thể kết nối với nhiều ngôn ngữ lập trình. Tức là có thể ánh xạ các mô hình UML về một ngôn ngữ lập trình như C++, Java...

Việc chuyển các mô hình trong UML thành Code trong ngôn ngữ lập trình 🡺 Forward engineering (Ex: Rational Rose)

* *Cung cấp tài liệu (Documenting)*

Giúp xây dựng tài liệu đặc tả - requirements

Tài liệu kiến trúc (architecture)

Tài liệu thiết kế

Source code

Tài liệu để kiểm thử - Test

Tài liệu mẫu - Prototype

Tài liệu triển khai – Deployment

**4. Mô hình và hướng nhìn**

Hướng nhìn chỉ ra những khía cạnh khác nhau của hệ thống cần phải được mô hình hóa. Một hướng nhìn không phải là một bản vẽ, mà là một sự trừu tượng hóa bao gồm một loạt các biểu đồ khác nhau. Chỉ qua việc định nghĩa của một loạt các hướng nhìn khác nhau, mỗi hướng nhìn chỉ ra một khía cạnh riêng biệt của hệ thống, người ta mới có thể tạo dựng nên một bức tranh hoàn thiện về hệ thống.

* *Biểu đồ*

Các biểu đồ góp phần chi tiết hóa hệ thống, mỗi biểu đồ là sự kết hợp của các thành phần cơ bản. Dựa và các biểu đồ, các nhà phát triển có thể giao tiếp với nhau và giải quyết các vấn đề phức tạp của hệ thống. Biểu đồ lớp là loại biểu đồ thông dụng nhất, dùng đề mô tả mối quan hệ giữa các lớp, giúp cho nhà phát triển có thể hiểu cấu trúc lớp của hệ thống

* *Hướng nhìn (View)*

Mô hình hóa một hệ thống phức tạp là một việc làm khó khăn khi xem xét một hệ thống, chúng ta cần xây dựng các mô hình từ những khía cạnh khác nhau, xuất phát từ thực tế là những người làm việc với hệ thống với những vai trò khác nhau sẽ nhìn hệ thống từ những khía cạnh khác nhau. Vì vậy một hệ thống thường được miêu tả trong một loạt các hướng nhìn khác nhau. Mỗi hướng nhìn sẽ thể hiện một bức ảnh ánh xạ của toàn bộ hệ thống và chỉ ra một khía cạnh riêng của hệ thống, Mỗi một hướng nhìn được miêu tả trong một loạt các biểu đồ, chứa đựng các thông tin nêu bật khía cạnh đặc biệt đó của hệ thống.

* **UML có tất cả các hướng nhìn sau:**
* *Hướng nhìn UseCase*

Mô tả ứng xử của hệ thống theo cách nhìn nhận của người dùng, người phân tích hệ thống.

Các thuật ngữ sử dụng trong hướng nhìn mang ý nghĩa rõ ràng về một lĩnh vực nào đó

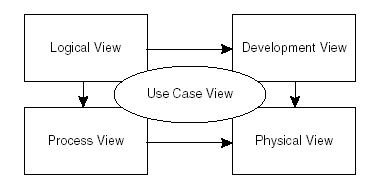
người dùng có thể kiểm tra xem các yêu cầu của mình đã được đáp ứng đầy đủ hay chưa hoặc có chức năng nào của hệ thống là không cần thiết.

Được miêu tả qua các biểu đồ Use case (use case diagram) và thỉnh thoảng cũng bao gồm cả các biểu đồ hoạt động (activity diagram). .

Hướng nhìn Use case mang tính trung tâm, bởi nó đặt ra nội dung thúc đẩy sự phát triển các hướng nhìn khác

* *Hướng nhìn thiết kế (Design)*
* Hướng nhìn thiết miêu tả phương thức mà các chức năng của hệ thống sẽ được cung cấp
* Chủ yếu nó được sử dụng cho các nhà thiết kế và nhà phát triển.
* Biểu đồ thường được sử dụng là biểu đồ lớp và biểu đồ tương tác
* *Hướng nhìn triển khai (Development)*
* Là một lời miêu tả của việc thực thi các modul cũng như sự phụ thuộc giữa chúng với nhau
* Nó thường được sử dụng cho nhà phát triển và thường bao gồm nhiều biểu đồ thành phần
* Hướng nhìn này mang tính tùy chọn
* *Hướng nhìn tiến trình (Process)*
* Chia hệ thống thành các tiến trình(process) và luồng(thread), mô tả việc đồng bộ hóa và các xử lý đồng thời.
* Các tiến trình và luồng thường được xác định dựa vào biểu đồ lớp thông qua các lớp động (active class)
* Bên cạnh việc chia hệ thống thành các tiểu trình có thể được thực thi song song, hướng nhìn này cũng phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các tiểu trình đó
* Bao gồm các biểu đồ động (trạng thái, trình tự, tương tác và hoạt động) cùng các biểu đồ thực thi (biểu đồ thành phần và biểu đồ triển khai)
* *Hướng nhìn vật lý (Physical)*
* Chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống
* Hướng nhìn triển khai giành cho các nhà phát triển, người tích hợp cũng như người thử nghiệm hệ thống và được thể hiện bằng các biểu đồ triển khai
* Ánh xạ các thành phần của hệ thống vào cấu trúc vật lý

Các hướng nhìn phải phù hợp với nhau vì cùng mô tả về một hệ thống. Có thể được sử dụng để kiểm tra tính hợp lệ giữa các hướng



Hình 1: Các hướng nhìn của UML

Danh sách các hướng nhìn là không giới hạn, phụ thuộc vào cách nhìn nhận hệ thống, ta có thể tạo ra một hướng nhìn mới

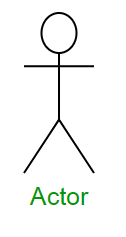
**Biểu đồ UML**

1. **Biểu đồ Use Case (Use Case Diagram)**

* Biểu đồ Use case tập trung vào các quy trình nghiệp vụ mà ứng dụng cần phải thực thi
* Biểu đồ Use case cho phép chúng ta cấu trúc hóa các toàn bộ ứng dụng xoay quanh quy trình cần phải thực thi
* Chỉ ra tập các tác nhân và Use case và mối quan hệ giữa chúng
* Biểu đồ Use case góp phần làm hiệu quả việc mô hình hóa

Các thành phần chính:

* Hệ thống: Với vai trò là thành phần của biểu đồ use case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta xây dựng.Một hệ thống ở trong biểu đồ use case không nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy,hoặc là một hệ thống thực như một doanh nghiệp, một trường đại học,…
* Tác nhân(actor):là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các use case. Một tác nhân có thể thực hiện nhiều use case và ngược lại một use case cũng có thể được thực hiện bởi nhiều tác nhân



Hình 2. Ký hiệu actor

* Các use case: Đây là thành phần cơ bản của biểu đồ use case. Các use case được biểu diễn bởi các hình elip.Tên các use case thể hiện một chức năng xác định của hệ thống.

Các use case được ký hiệu bằng hình elips:

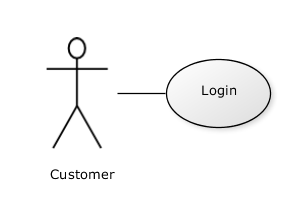
Hình 3. Ký hiệu use case

* Mối quan hệ giữa các use case:
* Association: thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau

Association

Hình 4. Ký hiệu Association

Ví dụ quan hệ Association:



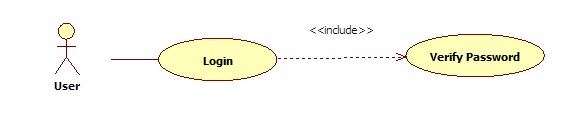
Hình 5. Quan hệ Association

* Include: là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.



Hình 6. Ký hiệu Include

Ví dụ quan hệ include:



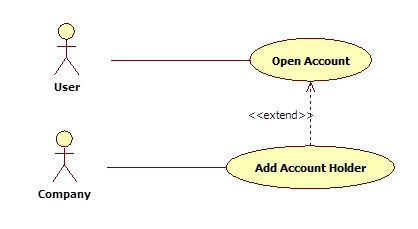
Hình 7. Quan hệ include

* Extent: Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.



Hình 8. Ký hiệu Extend

Ví dụ quan hệ extend:



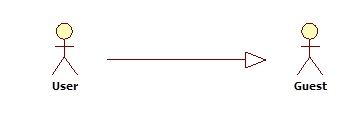
Hình 9. Quan hệ extend

* Generalization: được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.



Hình 10. Ký hiệu Generalization

Ví dụ quan hệ Generalization:



Hình 11. Quan hệ Generalization

### Biểu đồ hoạt động (Activity Diagram)

### Biểu đồ hoạt động (Activity Diagram) được sử dụng để mô tả các hoạt động và các hành động được thực hiện trong một use case.

### Các thành phần của biểu đồ hoạt động:

* Trạng thái khởi tạo hoặc điểm bắt đầu (Initial State or Start Point)



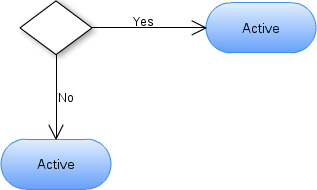
* Hoạt động hoặc trạng thái hoạt động (Activity or Action State)



Hoạt động và sự chuyển đổi hoạt động được ký hiệu và cách sử dụng hoàn toàn giống như trạng thái trong biểu đồ trạng thái đã nêu ở trên.

* Nút quyết định và rẽ nhánh

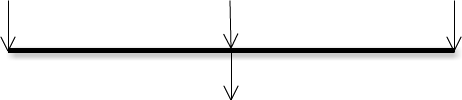
Nút rẽ nhánh trong biểu đồ hoạt động được kí hiệu bằng hình thoi màu trắng.



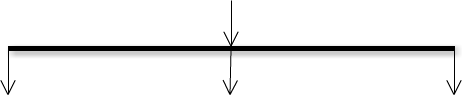
* Thanh tương tranh hay thanh đồng bộ

Có thể có nhiều luồng hành động được bắt đầu thực hiện hay kết thúc đồng thời trong hệ thống.

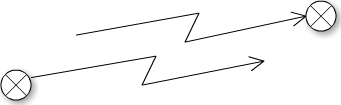
Thanh đồng bộ kết hợp:



Thanh đồng bộ chia nhánh:



* Cạnh gián đoạn (Interrupting Edge)



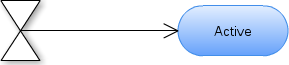
* Luồng hoạt động (Action Folow)



* Phân làn (Swimlanes)

Phân làn trong biểu đồ sử dụng là những đường nét đứt thẳng đứng theo các đối tượng. Phần kí hiệu này thường được sử dụng để làm rõ luồng hoạt động của các đối tượng riêng biệt.

* Thời gian sự kiện (Time Event)



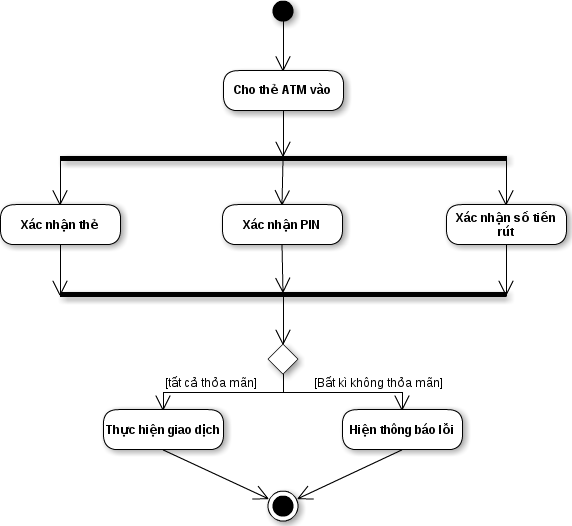
* Gửi và nhận tín hiệu (Sent and Received Signals)



* Trạng thái kết thúc hoặc điểm cuối (Final State or End Point)



**Ví dụ:**



### Như trên hình vẽ ta thấy có ba hoạt động cùng diễn ra là xác nhận thẻ, xác nhận mã số PIN và xác nhận số tiền rút.Chỉ khi sử dụng biểu đồ hoạt động mới có thể miêu tả được các hoạt động song song như vậy.

### Biểu đồ tương tác (Interaction Diagrams)

### Biểu đồ cộng tác (Collaboration Diagram)

### Là một dạng của biểu đồ tương tác mô tả cách thức tổ chức các đối tượng gửi và nhận các thông điệp

### Xét về mặt ngữ nghĩa *gần* giống như biểu đồ trình tự

### Khác với biểu đồ trình tự là ở đây tập trung vào ngữ cảnh và không gian thực hiện công việc

### Biểu đồ cộng tác mô tả cách tổ chức các tương tác của đối tượng và sự kết nối của chúng với đối tượng khác

### 

### Hình 12. Ví dụ biểu đồ cộng tác

### Biểu đồ trình tự (Sequence Diagram)

### Biểu đồ trình tự là biểu đồ theo thứ tự thời gian

### Đọc biểu đồ từ đỉnh xuống đáy

### Đọc biểu đồ bằng quan sát các đối tượng và thông điệp

### Mỗi đối tượng có vòng đời (Lifeline)

### Bắt đầu khi hình thành đối tượng, kết thúc khi phá hủy đối tượng

### Thông điệp được vẽ giữa hai đối tượng – thể hiện đối tượng gọi hàm đối tượng khác

### Thông điệp phản thân

### Ví dụ: Thêm hồ sơ sinh viên

### 

Hình 13. Biểu đồ trình tự

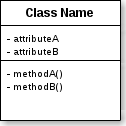
1. **Biểu đồ trạng thái (Statechart Diagram)**

* Biểu đồ trạng thái mô tả:
  + - * Quá trình sống của một lớp
      * Các sự kiện nào gây ra sự thay đổi từ trạng thái này đến trạng thái khác
      * Các hành động là kết quả của sự thay đổi trạng thái
* Biểu đồ dịch chuyển trạng thái được tạo cho các đối tượng có hành vi không cố định

1. **Biểu đồ lớp (Class Diagram)**

Một biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống. Các lớp là đại diện cho các “đối tượng” được xử lý trong hệ thống. Các lớp có thể quan hệ với nhau trong nhiều dạng thức:

* Liên kết (associated - được nối kết với nhau),
* Phụ thuộc (dependent - một lớp này phụ thuộc vào lớp khác),
* Chuyên biệt hóa (specialized - một lớp này là một kết quả chuyên biệt hóa của lớp khác),
* Hay đóng gói ( packaged - hợp với nhau thành một đơn vị).
* Một lớp có những thành phần sau:
* Tên lớp
* Các thuộc tính
* Các phương thức



Hình 14. Mô hình lớp

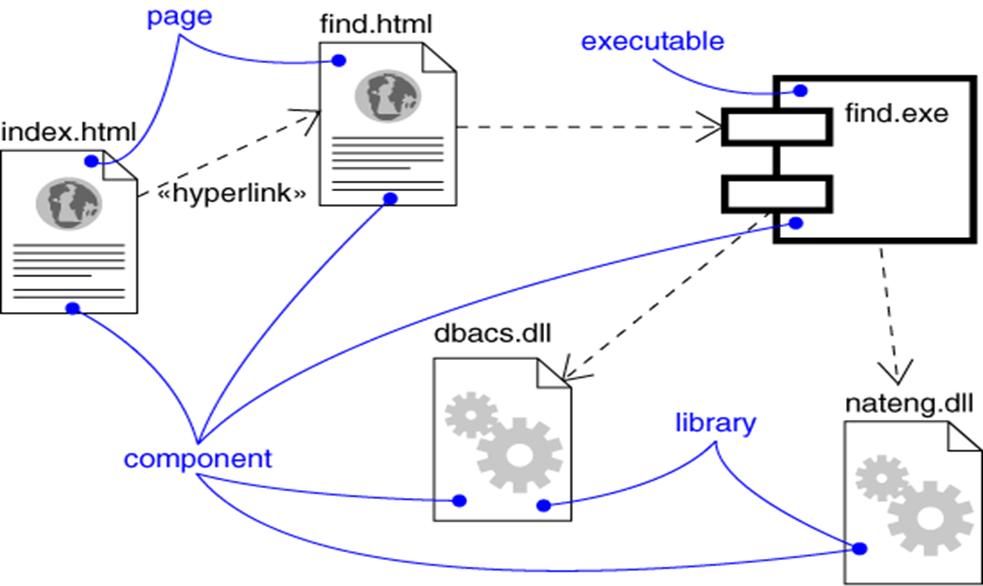
* Thuộc tính
* Thuộc tính cho biết cấu trúc của lớp
* Thuộc tính có thể được xác định thông qua địnhg nghĩa lớp, các yêu cầu và các tri thức về lĩnh vực phân tích
* Phương thức
* Hành vi của lớp được thể hiện thông qua các phương thức
* Các phương thức có thể xác định thông qua sự tương tác giữa các biểu đồ

1. **Biểu đồ đối tượng** **(Object Diagram)**

* Cung cấp một bức tranh thực tế về hệ thống nhằm mô tả các mối quan hệ tĩnh giữa các đối tượng
* Một biểu đồ đối tượng vì vậy là một ví dụ của biểu đồ lớp, chỉ ra một bức tranh thực tế có thể xảy ra khi hệ thống thực thi: bức tranh mà hệ thống có thể có tại một thời điểm nào đó.
* Rất hữu ích trong việc bổ sung các thể hiện của các mối quan giữa các đối tượng, góp phần kiểm tra tính đúng đắn của biểu đồ lớp
* Không được sử dụng nhiều

1. **Biểu đồ thành phần (Component Diagram)**

* Nhấn mạnh các mối quan hệ tĩnh giữa các thành phần phần mềm thực thi
* Đó có thể là các loại file .exe, .dll, .ocx, jar files, hay là Enterprise JavaBeans.
* Có thể được sử dụng để chỉ ra các phần mềm bên trong ứng dụng
* Các thành phần không phải là các lớp
* Mô tả cấu trúc vật lý của quá trình thực thi



Hình 15. Biểu đồ thành phần

1. **Biểu đồ triển khai (Deployment Diagram)**

* Biểu đồ triển khai chỉ ra cấu hình các phần tử xử lý lúc chương trình chạy, các nút trên mạng và các tiến trình phần mềm thực hiện trên những phần tử đó
* Mô tả mối quan hệ của hệ thống phần cứng
* Mô tả cấu trúc vật lý của hệ thống
* Biểu đồ triển khai chỉ ra toàn bộ các nút trên mạng, kết nối giữa chúng và các tiến trình chạy trên chúng
* Rất hữu ích đối với các hệ thống có môi trường cấu hình phức tạp
* Đối với các ứng dụng triển khai trên nhiều server và vùng lãnh thổ, việc sử dụng biểu đồ trển khai là cần thiết

**Chương trình vẽ UML**

1. **Giới thiệu về công cụ vẽ UML**

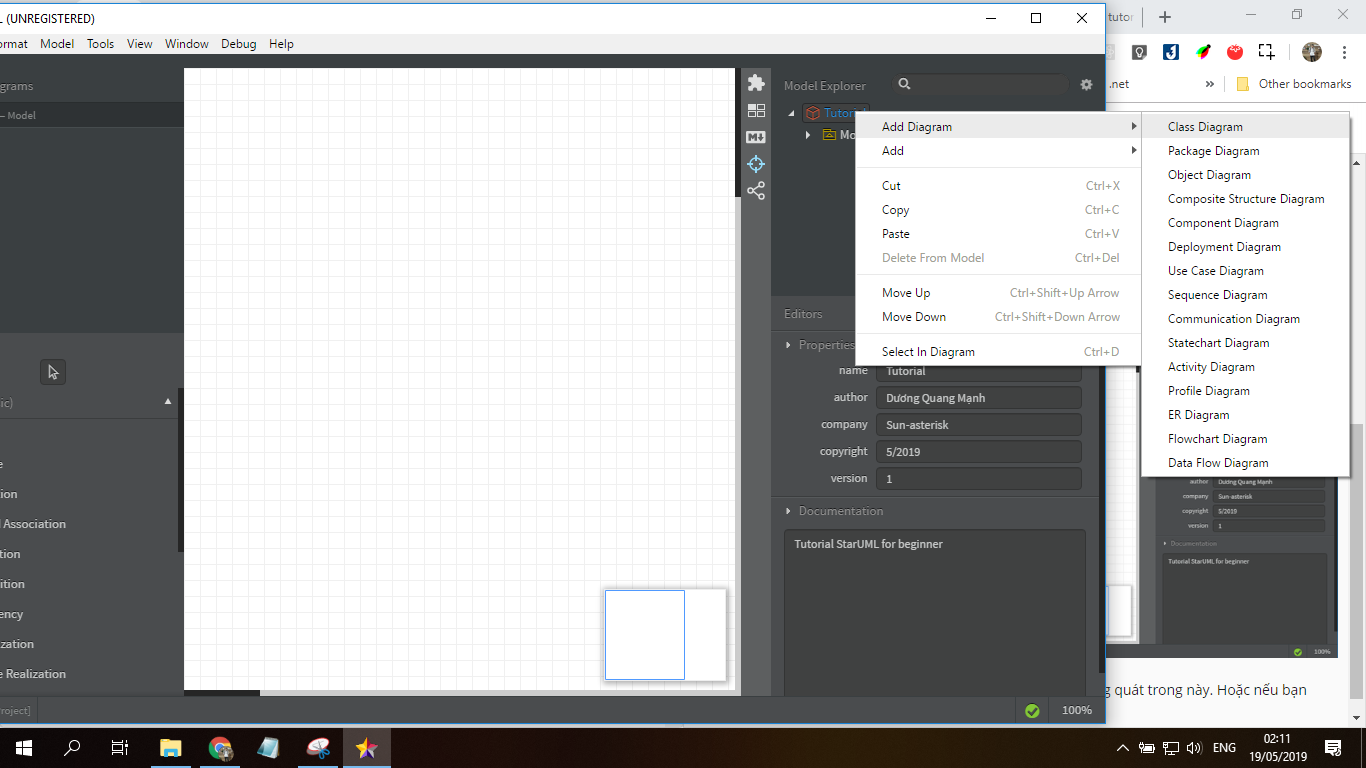
* Có rất nhiều công cụ được sử dụng để vẽ sơ đồ UML chuyên nghiệp như: Microsoft Visio, StarUML, Dia, ThinkComposer,…
* Các công cụ vẽ UML có cách sử dụng gần giống nhau và ký hiệu của các bản vẽ trên UML cũng đã thống nhất nên khi bạn đã nắm bắt một công cụ vẽ UML thì khi chuyển sang làm việc với một công cụ khác là không quá khó khăn.
* Trong bài này, nhóm mình sẽ trình bày về chương trình sẽ UML bằng phần mềm StarUML. StarUML là một phần mềm với mã nguồn mở, miễn phí và có đầy đủ các chức năng thích hợp cho việc vẽ sơ đồ UML.
* StarUML dùng để hỗ trợ phân tích và thiết kế hệ thống trong các dự án.

1. **Các chức năng của StarUML**

* Mô hình tiêu chuẩn UML chính xác: StarUML tuân thủ việc đặc tả UML tiêu chuẩn theo quy định của OMG cho mô hình phần mềm.
* Mở rộng tuyệt vời: tất cả chức năng của công cụ StarUML được tự động theo Microsoft COM. Bất kỳ ngôn ngữ hỗ trợ COM ( Java Script, VB, Delphi, C + +, C #, VB.NET, Python,…) có thể được dùng để kiểm soát StarUML.
* Hữu ích Add-Ins: Nó tạo ra mã nguồn trong các ngôn ngữ lập trình và chuyển đổi mã nguồn thành các mô hình.
* Chức năng xác minh mô hình phần mềm: StarUML có thể xác minh các mô hình phần mềm được xây dựng bởi người dùng, nhằm tạo điều kiện thuận lợi cho việc phát hiện sớm các lỗi.

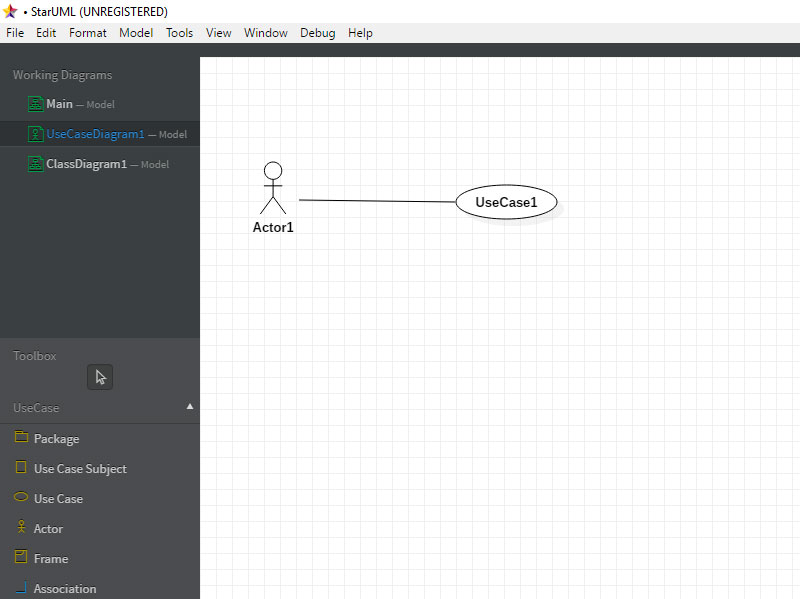
1. **Sử dụng StarUML**

* Để sử dụng StarUML, trước tiên bạn cần download phần mềm này tại trang chủ của nó (<http://staruml.io/download>).
* Sau khi cài đặt xong, bạn chỉ cần mở StarUML lên và vẽ các sơ đồ UML của bạn.



Hình 16: Giao diện phần mềm StarUML

* Khi mở StarUML lên, nó sẽ tạo sẵn cho bạn một model main và có thể thiết kế được các sơ đồ tổng quát bên trong. Nếu bạn muốn tạo một sơ đồ khác chỉ cần click chuột phải vào tên project chọn Add Diagram 🡪 Chọn các Diagram bạn muốn tạo.



Hình 17: Tạo bản vẽ bằng StarUML

* Sau khi bạn nhấn chọn Add Diagram 🡪 Use Case Diagram, nó sẽ hiện lên một bản thiết kế màu trắng và cửa sổ bên trái sẽ hiển thị thanh công cụ chứa các ký hiệu tương ứng của bản vẽ. Bạn có thể kéo các ký hiệu này vào bên trong trang màu trắng và bắt đầu thiết kế sơ đồ của bạn.
* StarUML hỗ trợ đầy đủ các khái niệm sơ đồ UML:
* Class Diagram
* Package Diagram
* Object Diagram
* Composite Structure Diagram
* Component Diagram
* Deployment Diagram
* Use Case Diagram
* Sequence Diagram
* Comunication Diagram
* Statechart Diagram
* Activity Diagram
* Profile Diagram
* ER Diagram
* Flowchart Diagram
* Data Flow Diagram
* Với mỗi Diagram khi tạo sẽ cho ra một model mới. Bằng cách kéo thả các thành phần tương ứng vào thì sẽ xây dựng được một Diagram hoàn chỉnh.

***Cách vẽ biểu đồ UML***

**1.Biểu đồ Use-case**

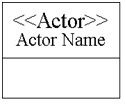
Một biểu đồ Use case chỉ ra một số lượng các tác nhân ngoại cảnh và mối liên kết của chúng đối với Use case mà hệ thống cung cấp. Một Use case là một lời miêu tả của một chức năng mà hệ thống cung cấp. Lời miêu tả Use case thường là một văn bản tài liệu, nhưng kèm theo đó cũng có thể là một biểu đồ hoạt động. Các Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao. Các Use case định nghĩa các yêu cầu về mặt chức năng đối với hệ thống.

* Hệ thống: Với vai trò là thành phần của biểu đồ use case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta xây dựng.Một hệ thống ở trong biểu đồ use case không nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy,hoặc là một hệ thống thực như một doanh nghiệp, một trường đại học,…
* Tác nhân(actor):là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các use case. Một tác nhân có thể thực hiện nhiều use case và ngược lại một use case cũng có thể được thực hiện bởi nhiều tác nhân

Tác nhân được ký hiệu:



Hoặc



Các use case: Đây là thành phần cơ bản của biểu đồ use case. Các use case được biểu diễn bởi các hình elip.Tên các use case thể hiện một chức năng xác định của hệ thống.

Các Use case được kí hiệu bằng hình elips.



Mối quan hệ giữa các use case:

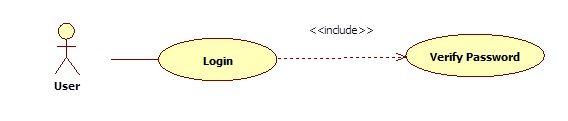
Association: thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau:

C:\Users\root\Desktop\d9e8ef3c-6812-4dda-a534-adcbaf4f3354.jpg

Ví dụ quan hệ association:



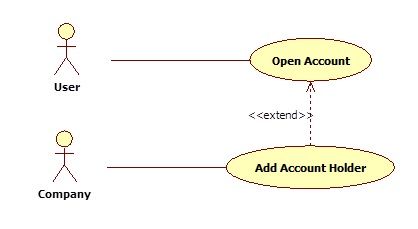
Ví dụ quan hệ include:



Extent: Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.



Ví dụ quan hệ extent:



Generalization: được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.

C:\Users\root\Desktop\f9b853ef-5876-4efa-b731-5dbb173f2feb.jpg

**2. Biểu đồ lớp**

Một biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống. Các lớp là đại diện cho các “đối tượng” được xử lý trong hệ thống. Các lớp có thể quan hệ với nhau trong nhiều dạng thức:

* liên kết (associated - được nối kết với nhau),
* phụ thuộc (dependent - một lớp này phụ thuộc vào lớp khác),
* chuyên biệt hóa (specialized - một lớp này là một kết quả chuyên biệt hóa của lớp khác),
* hay đóng gói ( packaged - hợp với nhau thành một đơn vị).

Tất cả các mối quan hệ đó đều được thể hiện trong biểu đồ lớp, đi kèm với cấu trúc bên trong của các lớp theo khái niệm thuộc tính (attribute) và thủ tục (operation). Biểu đồ được coi là biểu đồ tĩnh theo phương diện cấu trúc được miêu tả ở đây có hiệu lực tại bất kỳ thời điểm nào trong toàn bộ vòng đời hệ thống.

Một hệ thống thường sẽ có một loạt các biểu đồ lớp – không phải bao giờ tất cả các biểu đồ lớp này cũng được nhập vào một biểu đồ lớp tổng thể duy nhất – và một lớp có thể tham gia vào nhiều biểu đồ lớp.

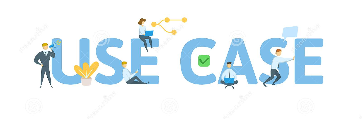
Một lớp có các thành phần sau

* Tên lớp
* Các thuộc tính
* Các phương thức

**Use Case Diagram**

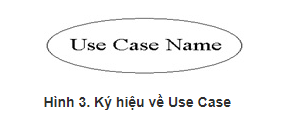
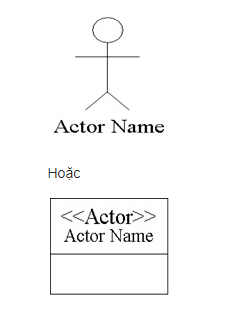
1. **Use case là gì?**

* Use case là một kỹ thuật được dùng trong kỹ thuật phần mềm và hệ thống để nắm bắt yêu cầu chức năng của hệ thống. Bao gồm các chức năng mong đợi của hệ thống (use case) và môi trường (actor) của nó.
* Mục đích: Là công cụ cho việc lên kế hoạch trong các dự án. Được dùng trong tất cả các giai đoạn của quy trình phát triển hệ thống.



1. **Thành phần của Use Case**

* Actor được dùng để chỉ người sử dụng hoặc một đối tượng nào đó bên ngoài tương tác với hệ thống chúng ta đang xem xét.
* Use Case là chức năng mà các Actor sẽ sử dụng.



1. **Relationshi p (Quan hệ)**

* Relationship hay còn gọi là conntector được sử dụng để kết nối giữa các đối tượng với nhau tạo nên bản vẽ Use Case. Có các kiểu quan hệ cơ bản sau:
* Association
* Generalization
* Include
* Extend
  1. **Quan hệ Association**
* Association thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau.
  1. **Quan hệ Generalization**
* Generalization được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.
  1. **Quan hệ Include**
* Include là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.
  1. **Quan hệ Extend**
* Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.

1. **Các giai đoạn xây dựng một Use Case Diagram**

* **Giai đoạn mô hình hóa:**

Bước 1: Thiết lập ngữ cảnh của hệ thống đích.

Bước 2: Chỉ định các Actor.

Bước 3: Chỉ định các Use Case.

Bước 4: Định nghĩa các quan hệ giữa các Actor và các Use Case.

Bước 5: Đánh giá các Actor và các Use Case để tìm cách chi tiết hóa.

* **Giai đoạn cấu trúc:**

Bước 6: Đánh giá các Use Case cho quan hệ phụ thuộc «include».

Bước 7: Đánh giá các Use Case cho quan hệ phụ thuộc «extend».

Bước 8: Đánh giá các Use Case cho quan hệ generalizations.

* **Giai đoạn review:**

Sau khi định nghĩa Use Case, cần tiến hành thử nghiệm Use Case:

+ Kiểm tra (verification): đảm bảo là hệ thống đã được phát triển đúng đắn và phù hợp với các đặc tả đã được tạo ra.

+ Phê chuẩn (validation): đảm bảo rằng hệ thống sẽ được phát triển chính là thứ mà khách hàng hoặc người sử dụng cuối thật sự cần đến.