Chương 1
Tổng quan về phân tích thiết kế hệ thống thông tin

Nguyễn Hồng Hạnh, MSc
Bộ môn Công nghệ phần mềm
Khoa CNTT&TT – ĐH Xây dựng Hà Nội
Email: hanhnh@huce.edu.vn

Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống hướng đối tượng
2. Tiến trình phát triển hệ thống
3. Giới thiệu phương pháp mô hình hóa

Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống
2. Tiến trình phát triển hệ thống
3. Giới thiệu phương pháp mô hình hóa

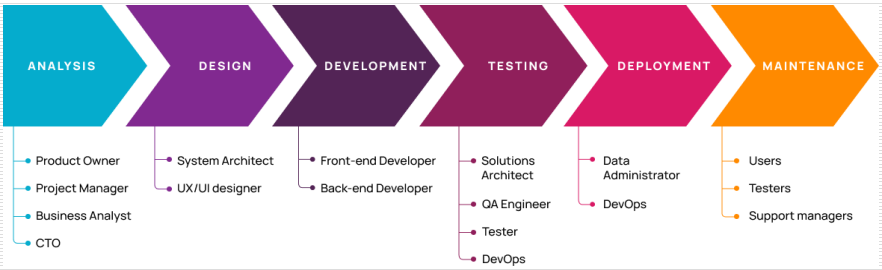
1.1 Tổng quan Phân tích thiết kế HT

- **Hệ thống? (HT)**
  - Hệ mặt trời, HT giao thông, HT luật pháp, HT tuần hoàn, HT thông tin,…
  - HT là một tập hợp gồm nhiều phần tử, có mối quan hệ ràng buộc lẫn nhau và cùng hoạt động hướng tới một mục đích chung
  - HT nhận cái vào (input) và xuất cái ra (output).
- **"Tại sao phải phân tích và thiết kế hệ thống thông tin?"**
  - Có cái nhìn đầy đủ, đúng đắn và chính xác về hệ thống thông tin được xây dựng trong tương lai.
  - Tránh sai lầm trong thiết kế, cài đặt.
  - Dễ sửa chữa, bổ sung và phát triển hệ thống trong quá trình sử dụng hoặc khi hệ thống yêu cầu.
  - Tăng vòng đời hệ thống.

1.1 Tổng quan Phân tích thiết kế HT

- **Tiến trình phát triển phần mềm (**Software Development Lifecycle **SDLC).**

Là một tập hợp các hoạt động để chuyển yêu cầu người sử dụng thành một hệ thống phần mềm

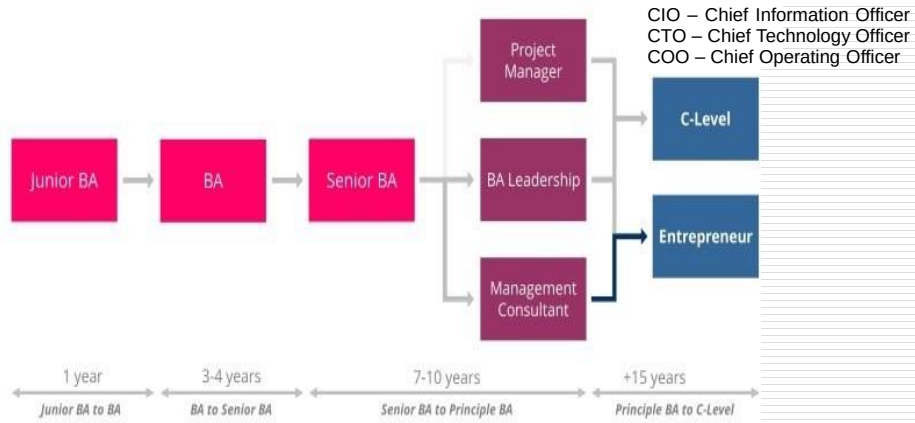


5

5

1.1 Tổng quan Phân tích thiết kế HT

- Quy trình phân tích nghiệp vụ (Business Analysis – Process)



CIO – Chief Information Officer
CTO – Chief Technology Officer
COO – Chief Operating Officer

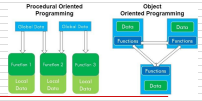
6

6

1.1 Tổng quan Phân tích thiết kế HT

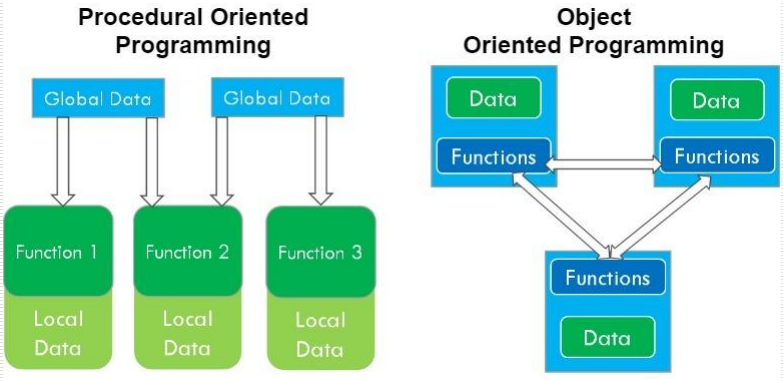
- **Có cần phương pháp trong quá trình PT TK HTTT?**
  - Làm thơ, soạn nhạc cũng cần có phương pháp!
  - Các phương pháp để phân tích và thiết kế phần mềm
    - Phân tích thiết kế HT theo hướng cấu trúc
    - Phân tích thiết kế HT theo hướng đối tượng

1.1 Tổng quan Phân tích thiết kế HT



- **Phương pháp hướng chức năng (có cấu trúc)**
  - nở rộ vào những năm 70, 80 thế kỷ XX
  - lấy chức năng làm đơn vị phân rã khi tiến hành phân tích HT.
  - cài đặt HT bằng các NNLT thủ tục: Pascal, C,…
  - nhược điểm: HT khó sửa chữa, khó nâng cấp, khó tái sử dụng. Với hệ thống có yêu cầu thường xuyên thay đổi
- **Phương pháp hướng đối tượng**
  - ra đời vào những năm 90 của thế kỷ XX
  - lấy đối tượng làm đơn nguyên cơ bản của HT
  - đối tượng: kết hợp giữa chức năng và dữ liệu
  - cài đặt bằng các NNLT HĐT: C++, Java, C#,…

1.1 Tổng quan Phân tích thiết kế HT



9

9

Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống
2. Tiến trình phát triển hệ thống
3. Giới thiệu phương pháp mô hình hóa

10

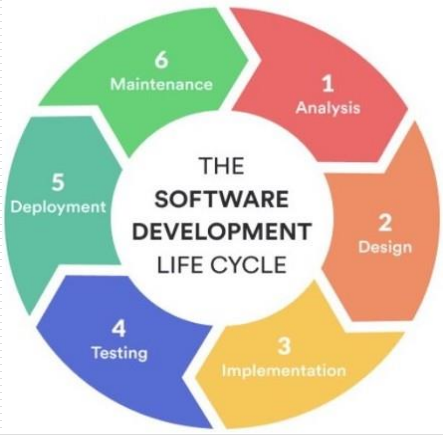
10

Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống
2. Tiến trình phát triển hệ thống
   - 1.2.1 Các giai đoạn trong vòng đời phần mềm
   - 1.2.2 Các mô hình phát triển phần mềm thông dụng
3. Giới thiệu phương pháp mô hình hóa

1.2 Tiến trình phát triển hệ thống

1.2.1 Các giai đoạn trong vòng đời phần mềm



1.2.1 Các giai đoạn trong vòng đời phần mềm

6 giai đoạn trong SDLC:

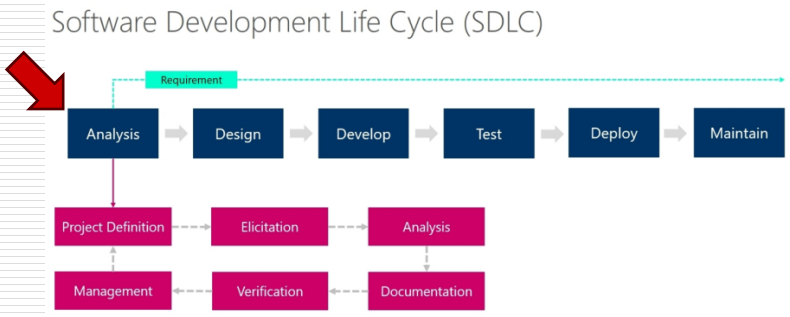
- **Analysis:** phân tích xem mình sẽ *làm những gì*
- **Design:** mình sẽ *thiết kế phần mềm như thế nào*
- **Develop:** mình sẽ *code ra sao*
- **Test:** phần mềm *được đem đi kiểm tra xem chạy đúng yêu cầu thiết kế chưa?*
- **Deploy:** phần mềm *được đưa vào sử dụng*
- **Maintain:** giai đoạn *bảo trì, hỗ trợ* khách hàng sử dụng phần mềm.

13

1.2.1 Các giai đoạn trong vòng đời phần mềm

1.2.1.1 Analysis

Là giai đoạn đội phát triển dự án sẽ phân tích để hiểu được vấn đề của khách hàng và những gì mà họ đang mong đợi. Giai đoạn xác định yêu cầu

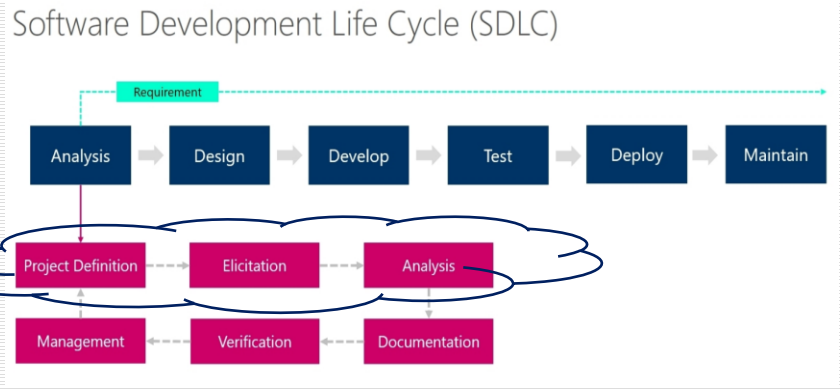


14

1.2.1 Các giai đoạn trong vòng đời phần mềm

1.2.1.1 Analysis

3 bước quan trọng trong giai đoạn Analysis



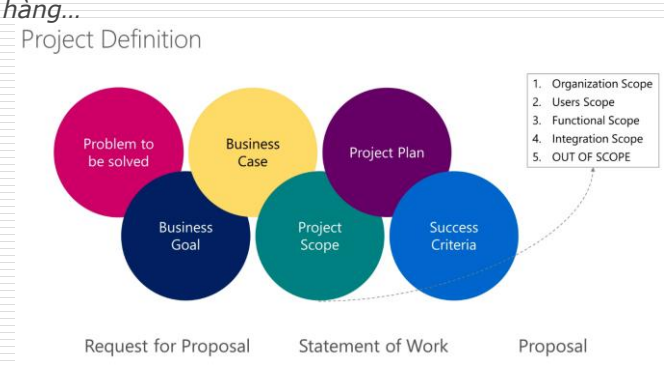
15

15

1.2.1.1 Analysis

A. Project Definition – Xác định dự án

Hiểu được dự án làm gì, khách hàng là ai, thuộc lĩnh vực gì, vấn đề của họ ra sao, hiện trạng hiện tại của họ như thế nào, mục tiêu hướng đến của họ là gì, phạm vi của dự án như thế nào, kế hoạch thực hiện dự án (time schedule) ra sao, những tiêu chí giúp dự án thỏa mãn kỳ vọng của khách hàng…



16

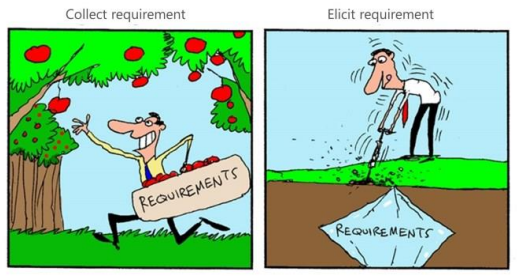
16

1.2.1.1 Analysis

B. Elicitation – Khai thác thông tin

Mục đích: Nắm được, khai thác được chính xác tất cả những yêu cầu (requirements) từ phía khách hàng (cả tường minh (stated) lẫn ngầm ẩn (unstated)).

Output: Thông tin, **rất nhiều thông tin,** tồn tại dưới nhiều dạng (ghi chép, hình ảnh, âm thanh…)



17

1.2.1.1 Analysis

C. Analysis – Phân tích thông tin

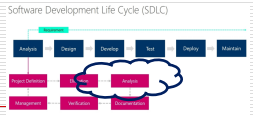
Mục đích: Hệ thống lại thông tin, đơn giản là **sắp xếp & phân loại** mọi thứ cho phù hợp, đẹp đẽ, sạch sẽ để dễ dàng nắm bắt**, hiểu được vấn đề** và **tìm hướng giải quyết**.

"Analysis means simply breaking down the information of an object, entity, process, or anything else to understand its functioning"

Sandhya Jane - tác giả cuốn Business Analysis: The question and answer book

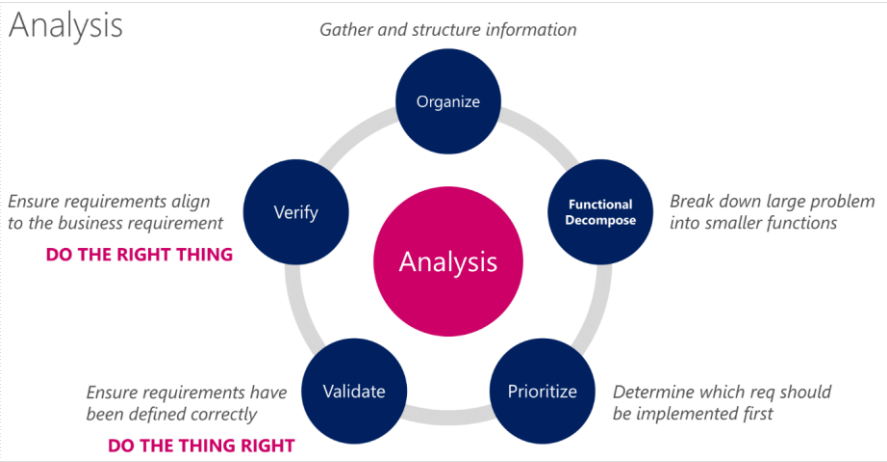
Nhận diện đúng đâu là yêu cầu thực sự khách hàng mong đợi ở hệ thống. Và trong những yêu cầu đó thì cái gì nên ưu tiên hàng đầu, cái gì họ cần nhất?

18



1.2.1.1 Analysis

C. Analysis – Phân tích thông tin



1.2.1.1 Analysis

D. Documentation – Tài liệu hóa

Mô tả chi tiết yêu cầu của khách hàng, để thực hiện thường áp dụng 2 cách sau:

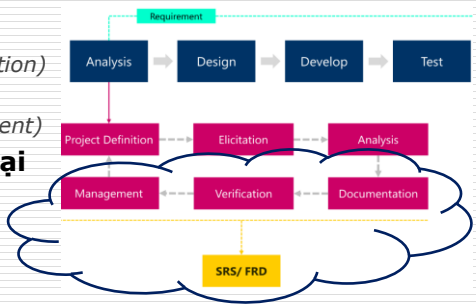
1. Tận dụng các *template* có sẵn
2. Dùng *ngôn ngữ mô hình hóa* để tài liệu hóa. Hai loại phổ biến nhất là BPMN và UML.

Tài liệu output:

SRS *(Software Requirement Specification)*

Hoặc

FRD *(Functional Requirement Document)*



E. Verification – Kiểm tra lại

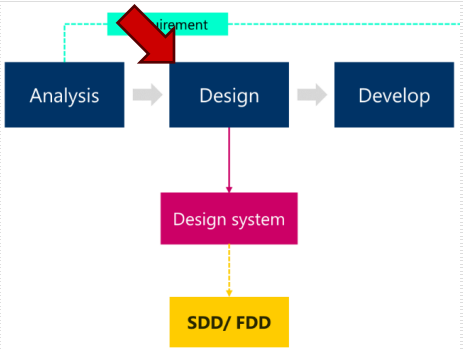
F. Management

1.2.1 Các giai đoạn trong vòng đời phần mềm

1.2.1.2 Design

Giai đoạn can thiệp sâu về kỹ thuật, bao gồm:

- Thiết kế Database
- Vẽ Mockup Giao diện
- Thiết kế UX/UI
- Thiết kế Business Process Flow
- Thiết kế bộ phân quyền hệ thống
- …

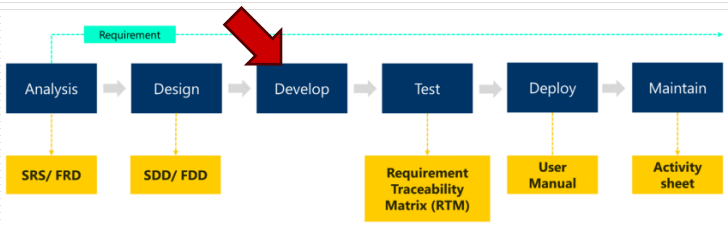


Tài liệu Output: Tài liệu thiết kế

- **SDD** *(Software Design Document)* hoặc
- **FDD** *(Functional Design Document)*

1.2.1 Các giai đoạn trong vòng đời phần mềm

1.2.1.3 Develop – Quá trình lập trình sản phẩm



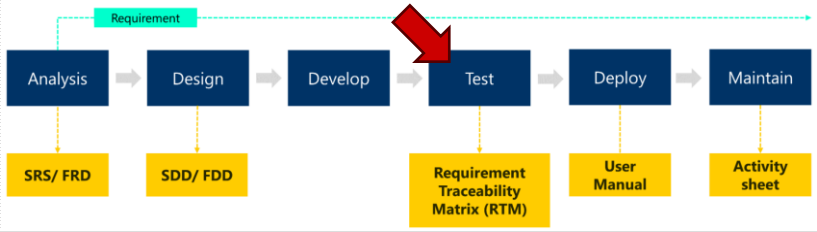
- Giai đoạn dài nhất của SDLC
- Nhiệm vụ được phân chia và giao cho các nhà phát triển (Developers)
- Các nhà phát triển bắt đầu viết mã, đồng thời viết các tình huống kiểm thử ở mức đơn vị (unit test), xây dựng và triển khai hệ thống trên một môi trường lựa chọn.

1.2.1 Các giai đoạn trong vòng đời phần mềm

1.2.1.4 Test –Kiểm thử

Internal Testing – test nội bộ đảm bảo Dev cài đặt đúng yêu cầu thiết kế, đảm bảo trả kết quả đúng cam kết với khách hàng. Sử dụng Requirement Traceability Matrix để theo dõi quá trình test case khớp với requirement (thành công hay thất bại)

External Testing – thực hiện **User Acceptance Test (UAT)** với khách hàng



1.2.1 Các giai đoạn trong vòng đời phần mềm

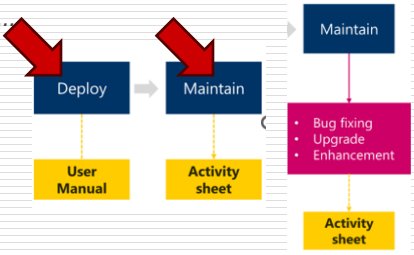
2.1.5 Deploy – Làm những việc để **hệ thống sẵn sàng được sử dụng**

- Build solution: *từ môi trường Dev lên môi trường Production.*
- Migrate data: *là chuyển toàn bộ data hiện tại của khách hàng từ hệ thống cũ sang hệ thống mới.*
- Thiết lập người dùng như: *phân quyền, cập nhật tài khoản, thông tin cá nhân…*
- ***Hướng dẫn người dùng sử dụng hệ thống*** *(User Manual)*

- có thể ở nhiều dạng: *tệp pdf, video, …*

2.1.6 Maintain – Bảo trì

- Sửa lỗi phát sinh
- Nâng cấp hệ thống



Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống
2. Tiến trình phát triển hệ thống
   - 1.2.1 Các giai đoạn trong vòng đời phần mềm
   - 1.2.2 Các mô hình phát triển phần mềm thông dụng
3. Giới thiệu phương pháp mô hình hóa

1.2.2. Các mô hình PTPM thông dụng

- Quy trình phần mềm về cơ bản gồm 6 giai đoạn, nhưng khi triển khai thực tế nó sẽ linh hoạt theo từng **phương pháp quản lý dự án** (project management methodology) tùy theo tính chất của từng dự án.
- Phương pháp quản lý dự án (hay còn gọi là mô hình phát triển phần mềm)

SDLC
Models

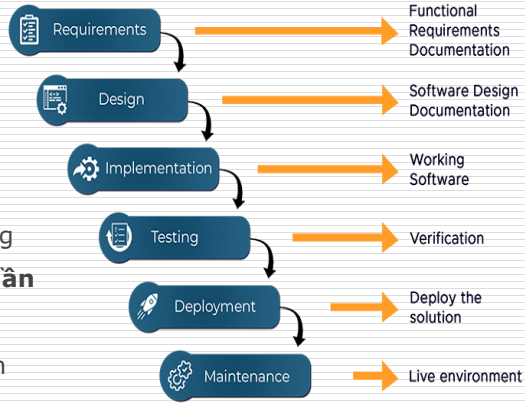
Waterfall

27

SDLC
Waterfall

Quy trình sẽ đi theo trình tự từ trái qua phải (không đi ngược lại), mỗi bước chỉ đi qua **một** lần.

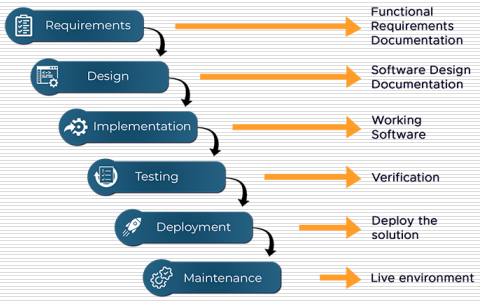
Phải đợi dự án chạy qua từng giai đoạn **từ đầu tới cuối 1 lần duy nhất** rồi **mới biết được thành phẩm cuối cùng** hình dạng ra sao.



28

SDLC
Waterfall

Ưu điểm	Nhược điểm
Đơn giản trong quản lý: Mỗi giai đoạn đều định nghĩa rõ kết quả và quy trình	Thiếu tính linh hoạt, quá cứng nhắc, không dễ dàng thay đổi
Dễ dàng phân loại và đánh giá độ ưu tiên cho các công việc	Rủi ro cao và không chắc chắn tạo ra được sản phẩm ổn định
Phù hợp với các dự án vừa và nhỏ với các yêu cầu rõ ràng	Không phù hợp với các dự án dài hơi, yêu cầu thay đổi thường xuyên
Tài liệu được làm cẩn thận rõ ràng	Kiểm thử chỉ xuất hiện ở khâu cuối, tăng khả năng rủi ro, tính đảm bảo chất lượng thấp
Phạm vi dự án, chi phí, và thời gian hoàn thành có khung rất rõ ràng vì vậy khách hàng nắm được chính xác về sản phẩm cuối cùng của họ	Việc tích hợp chỉ thực hiện tại giai đoạn cuối, không cho phép chọn lựa phát triển các tính năng nâng cao

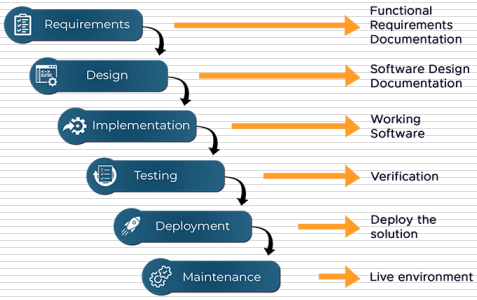


29

SDLC
Waterfall

Áp dụng các hệ thống/ dự án:

- Các dự án vừa hoặc nhỏ đơn giản với các yêu cầu được xác định rõ ràng và không/ ít thay đổi (ví dụ: phát triển trang web của công ty nhỏ)
- Với nhu cầu về ngân sách và mốc thời gian có thể dự đoán được (ví dụ: các dự án của chính phủ)
- Phải tuân thủ nhiều quy tắc và quy định (ví dụ: dự án chăm sóc sức khỏe)



30

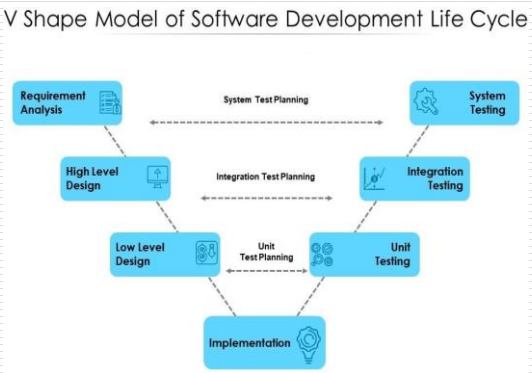
SDLC
Models

V- shaped

31

SDLC
V-shaped

- Được gọi là mô hình xác nhận, một phần mở rộng hơn của mô hình thác nước
- Mô hình V là một mô hình tuyến tính với mỗi giai đoạn có một hoạt động kiểm nghiệm tương ứng
- Lập kế hoạch kiểm thử từ sớm

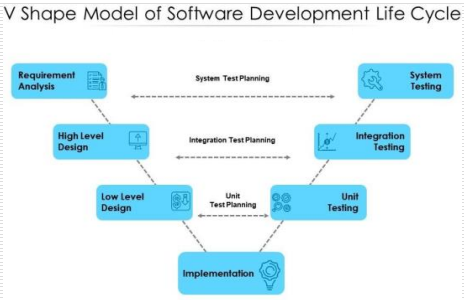


32

SDLC

V-shaped

Ưu điểm	Nhược điểm
Các lỗi trong đặc tả yêu cầu, lỗi trong lập trình và kiến trúc có thể được phát hiện sớm	Chi phí rất đắt và tốn nhiều thời gian phát triển
Kiểm thử chặt chẽ nên phát hiện bug rất nhanh	Các thay đổi trong suốt quá trình phát triển rất đắt và tốn thời gian, việc triển khai cũng rất khó khăn, chi phí thường bị đội lên rất cao
Phù hợp với các dự án cỡ nhỏ khi mà các yêu cầu rất rõ ràng và ít thay đổi	Thiếu tính linh hoạt

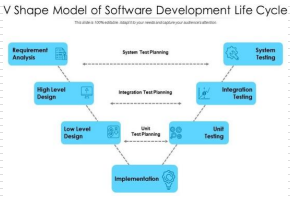


33

SDLC

V-shaped

Sử dụng trong các hệ thống/ dự án:



- Các dự án mà sự cố và thời gian ngừng hoạt động là không thể chấp nhận được (ví dụ: phần mềm y tế, quản lý đội bay hàng không, phần mềm giao dịch).
- Mô hình chữ V phù hợp với các loại dự án tương tự như Waterfall.
- Đối với các dự án yêu cầu thử nghiệm sản phẩm chính xác
- Đối với các dự án vừa và nhỏ, nơi các yêu cầu được xác định trước nghiêm ngặt

34

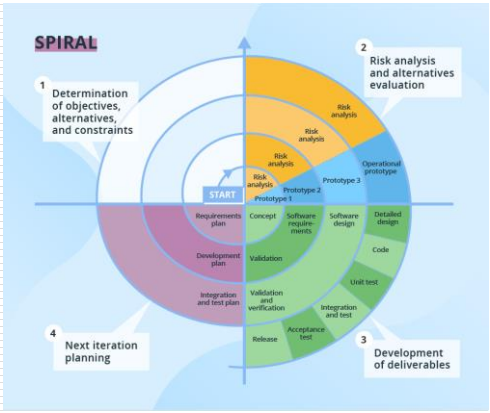
SDLC
Models

Spiral

35

SDLC
Spiral

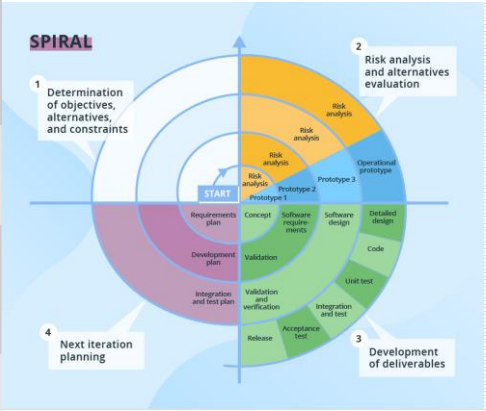
- Kết hợp xây dựng kiến trúc và tạo bản mẫu theo từng giai đoạn
- Kế hoạch phát triển được đưa ra dựa trên dữ liệu thống kê, nhận được từ kết quả của các dự án trước đó hoặc theo kinh nghiệm của nhà phát triển.
- Việc chuyển sang giai đoạn tiếp theo được thực hiện theo kế hoạch, ngay cả khi công việc ở giai đoạn trước chưa được thực hiện xong



36

SDLC
Spiral

Pros	Cons
Vòng đời được chia thành các phần nhỏ và nếu tỷ lệ rủi ro cao hơn, giai đoạn phát triển có thể được hoàn thành sớm hơn nhằm định vị các giải pháp xử lý	Chi phí khá đắt
Quy trình phát triển được làm tài liệu khá chính xác nhưng vẫn có thể mở rộng nếu có thay đổi	Việc kiểm soát rủi ro đòi hỏi sự tham gia của các chuyên gia có tay nghề cao
Khả năng mở rộng cho phép thực hiện các thay đổi và thêm tính năng mới ngay cả ở giai đoạn tương đối muộn	Không hiệu quả với các dự án nhỏ
Các bản mẫu được tạo ra sau mỗi vòng phát triển nên người dùng sớm có thể chỉ ra được các sai sót	Có số lượng lớn các giai đoạn trung gian vì vậy đòi hỏi quá nhiều tài liệu

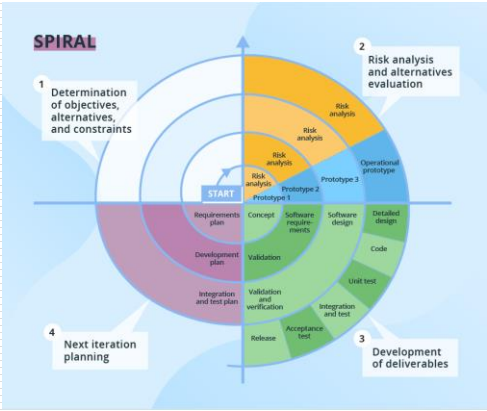


37

SDLC
Spiral

Sử dụng trong các dự án:

- Khách hàng không chắc chắn về các yêu cầu
- Các chỉnh sửa lớn được mong đợi trong chu kỳ phát triển
- Các dự án có rủi ro trung bình hoặc cao, trong đó điều quan trọng là phải ngăn chặn những rủi ro này
- Sản phẩm mới nên được phát hành trong một vài giai đoạn để có đủ phản hồi của khách hàng



38

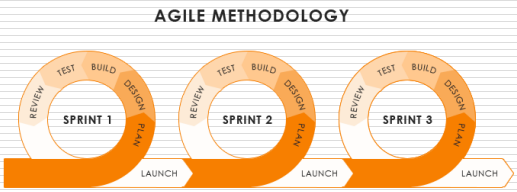
SDLC
Models

Agile

39

SDLC
Agile

- Hơn 70% các tổ chức sử dụng phương pháp tiếp cận Agile trong các dự án CNTT của mình
- Các tên khác (subtypes): Scrum, Extreme Programming và Kanban, Lean...



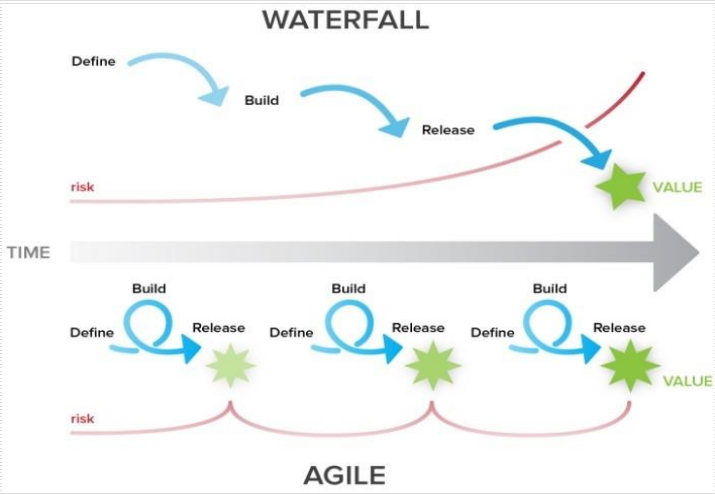
Sử dụng trong các dự án:

•Yêu cầu của người dùng thay đổi thường xuyên

•Hầu hết các dự án quy mô cỡ vừa

•Phát triển phần mềm có khả năng linh hoạt tùy biến cao vì các yêu cầu nghiệp vụ chưa rõ ràng, chưa thể chuyển một cách chắc chắn sang các yêu cầu chi tiết khi xây dựng hệ thống vì có thể còn thay đổi

40



41

SDLC

Agile

Ưu điểm	Nhược điểm
✓ Phương pháp tiếp cận theo hướng thích ứng (Agile) giúp phản hồi nhanh với sự thay đổi	▪ Focuses on working with software and lacks documentation efficiency
✓ Khách hàng và nhóm phát triển trao đổi với nhau hàng ngày	
✓ Việc sửa chữa các yêu cầu chức năng được thực hiện ngay trong quá trình phát triển sản phẩm để tăng khả năng cạnh tranh	▪ Khó khăn trong việc tính toán chi phí cuối cùng vì những thay đổi liên tục
✓ Dự án được chia theo các vòng lặp ngắn và minh bạch.	▪ Nhóm phải rất chuyên nghiệp và hướng đến khách hàng (và có kỹ năng ước tính công việc, kỹ năng đàm phán và kỹ năng giao tiếp)
✓ Rủi ro được giảm thiểu nhờ quy trình thay đổi linh hoạt	▪ Những yêu cầu mới có thể xung đột với kiến trúc hệ thống hiện tại
✓ Sản phẩm được phát hành sớm và thường xuyên cập nhật	▪ Với tất cả các chỉnh sửa và thay đổi, có khả năng dự án sẽ vượt quá thời gian dự kiến

42

Nội dung chính

1. Tổng quan về phân tích và thiết kế hệ thống
2. Tiến trình phát triển hệ thống
   - 1.2.1 Các giai đoạn trong vòng đời phần mềm
   - 1.2.2 Các mô hình phát triển phần mềm thông dụng
3. Giới thiệu phương pháp mô hình hóa

1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML

1.3 Giới thiệu phương pháp mô hình hóa

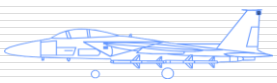
1.3.1 Khái niệm mô hình và mô hình hóa

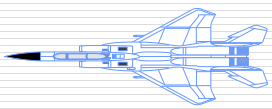
1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML

1.3.1 Khái niệm mô hình và mô hình hóa

- **Mô hình:** là một dạng trừu tượng hóa/một hình ảnh/một biểu diễn của một hệ thống thực, được diễn tả:
  - ở một mức độ trừu tượng hóa nào đó
  - theo một quan điểm/góc nhìn nào đó
  - bởi một hình thức diễn tả hiểu được nào đó như văn bản, đồ thị, phương trình,…





1.3.1 Khái niệm mô hình và mô hình hóa

- **Mô hình hóa (MHH):** là quá trình dùng mô hình để nhận thức và diễn tả một hệ thống.

Quá trình PT&TK hệ thống cũng còn được gọi là quá trình mô hình hóa hệ thống.

"Một bức tranh nói nhiều hơn cả ngàn từ".

Ngạn ngữ

47

47

1.3.1 Khái niệm mô hình và mô hình hóa

- Mục đích mô hình hóa:
  - Để hiểu
  - Để trao đổi
  - Để hoàn chỉnh
- MHH tốt phải thỏa các yêu cầu sau:
  - dễ đọc
  - dễ hiểu (understandable)
  - dễ trao đổi
  - chính xác (accurate)
  - chặt chẽ, đồng nhất (consistent)
  - đầy đủ
  - dễ thực hiện, dễ thay đổi (changeable)

48

48

1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML

1.3.2 Phương pháp mô hình hóa

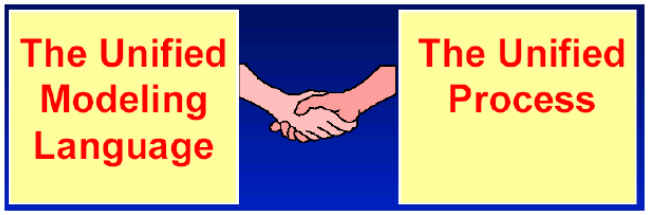
- Kết hợp 3 thành phần:
  - Hệ thống các ký pháp (notation) bao gồm các khái niệm và ký pháp mô hình tương ứng
  - Một tiến trình gồm các bước cần tiến hành, các sản phẩm (tư liệu, mô hình) qua từng giai đoạn, cách điều hành tiến trình, cách đánh giá chất lượng
  - Công cụ hỗ trợ (CASE): phần mềm hỗ trợ cho quá trình MHH, có khả năng:
    - sản sinh các MH và biểu đồ
    - biến đổi và điều chỉnh nhanh các MH và biểu đồ
    - kiểm tra cú pháp, sự chặt chẽ, đầy đủ
    - kiểm thử và đánh giá
    - mô phỏng và thực hiện mô hình

1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML



1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML

- 1.3.3.1 UML là gì?
- 1.3.3.2 Lịch sử ra đời và phát triển của UML
- 1.3.3.3 Các miền ứng dụng của UML
- 1.3.3.4 OOAD sử dụng UML như thế nào?
- 1.3.3.5 Các loại biểu đồ trong UML

1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

1.3.3 Giới thiệu UML

- 1.3.3.1 UML là gì?
- 1.3.3.2 Lịch sử ra đời và phát triển của UML
- 1.3.3.3 Các miền ứng dụng của UML
- 1.3.3.4 OOAD sử dụng UML như thế nào?
- 1.3.3.5 Các loại biểu đồ trong UML

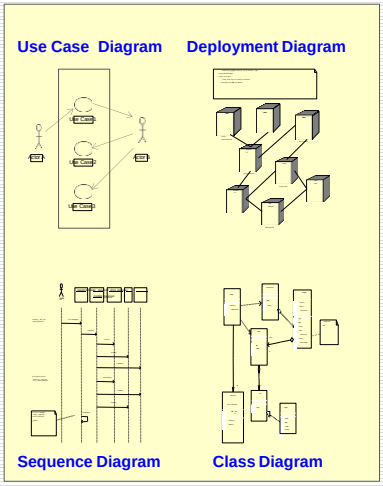
1.3.3.1.UML là gì?

- Ngôn ngữ mô hình hóa thống nhất ***(Unified Modeling Language - UML)***
- UML là ngôn ngữ để:
  - Trực quan hóa (visualizing)
  - Đặc tả (specifying)
  - Xây dựng (constructing)
  - Tài liệu hóa (documenting)

Các thành phần cấu thành (artifact) của một hệ thống

1.3.3.1.UML là gì?

- Nói một cách đơn giản UML dùng để tạo ra các bản vẽ nhằm mô tả thiết kế hệ thống.
- Các bản vẽ này được sử dụng để các nhóm thiết kế trao đổi với nhau cũng như dùng để thi công hệ thống (phát triển), thuyết phục khách hàng, các nhà đầu tư v.v..



55

55

1.3 Giới thiệu phương pháp mô hình hóa

56

56

1.3.3.2. Lịch sử phát triển của UML

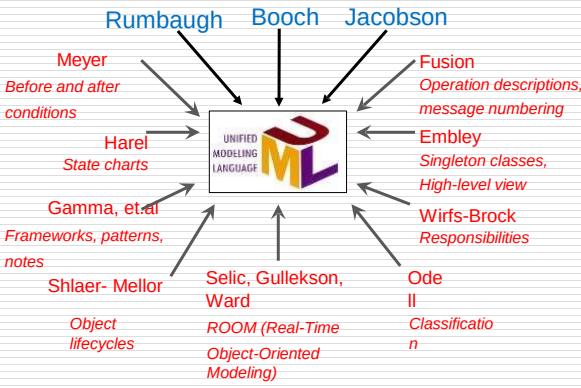
- Vào 1994, có hơn 50 phương pháp mô hình hóa hướng đối tượng:
  - Fusion, Shlaer-Mellor, ROOM, Class-Relation,Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS ...
- "Meta-models" tương đồng với nhau
- Các ký pháp đồ họa khác nhau
- Quy trình khác nhau hoặc không rõ ràng
- Cần chuẩn hóa và thống nhất các phương pháp

57

1.3.3.2. Lịch sử phát triển của UML

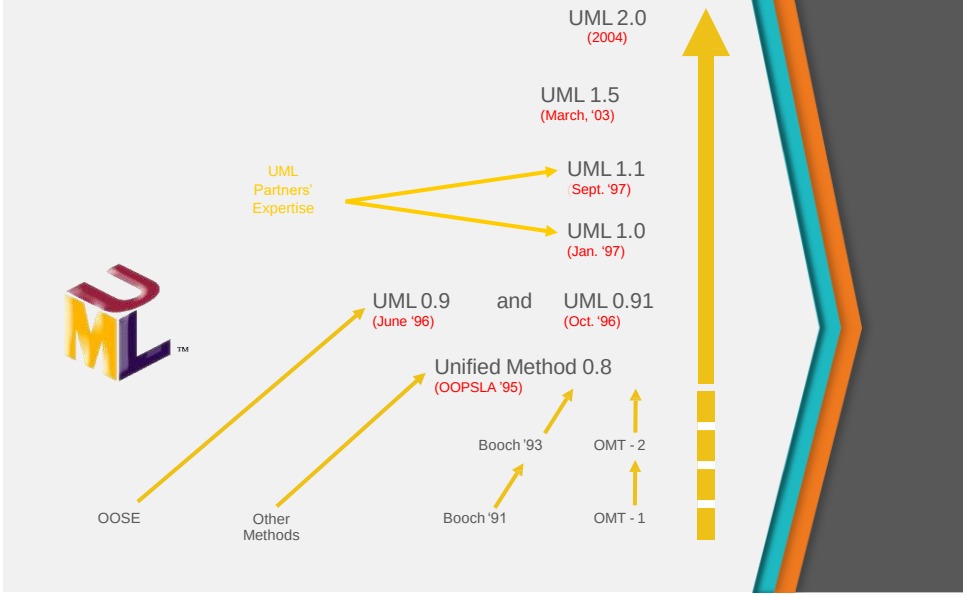
UML là ngôn ngữ hợp nhất các mô hình khác nhau.

UML được công nhận là chuẩn chung vào năm 1997.



58

1.3.3.2. Lịch sử phát triển của UML



59

1.3 Giới thiệu phương pháp mô hình hóa

60

1.3.3.3 Các miền ứng dụng của UML

- Hệ thống thông tin (Information System)
- Hệ thống kỹ thuật (Technical System)
- Hệ thống nhúng (Embedded System)
- Hệ thống phân tán (Distributed System)
- Hệ thống giao dịch (Business System)
- Phần mềm hệ thống (System Software)

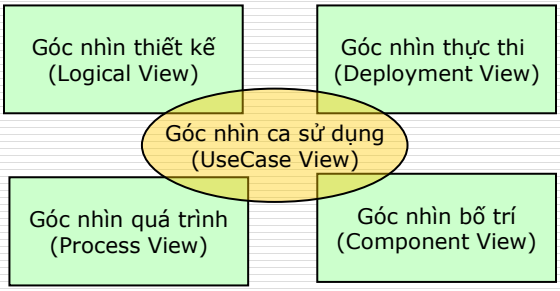
1.3 Giới thiệu phương pháp mô hình hóa

1.3.3.4 OOAD sử dụng UML như thế nào

- OOAD (Object Oriented Analysis and Design): *cần các bản vẽ để mô tả hệ thống được thiết kế.*
- UML là ngôn ngữ đặc tả các bản vẽ: *cần nội dung thể hiện.*
- Do vậy, chúng ta phân tích và thiết kế hệ thống theo hướng đối tượng (xem hệ thống gồm những đối tượng sống trong đó và tương tác với nhau) và sử dụng UML để biểu diễn các thiết kế để từ đó xây dựng và tạo ra hệ thống.

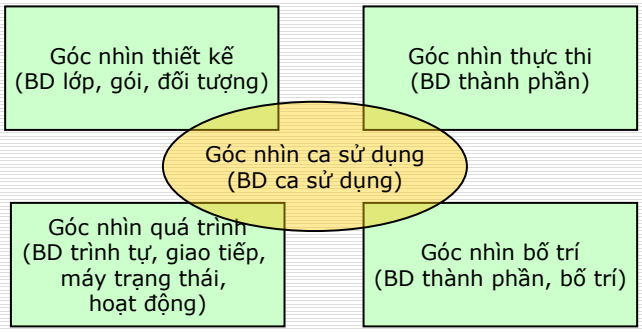
1.3.3.4 OOAD sử dụng UML như thế nào

- UML cung cấp các biểu đồ để diễn tả hệ thống (HT)
- Mỗi biểu đồ chỉ có thể diễn tả HT theo một **góc nhìn (view)** nhất định.
- Trong phần mềm, OOAD sử dụng UML dưới các góc nhìn sau



1.3.3.4 OOAD sử dụng UML như thế nào

- Mỗi góc nhìn biểu diễn bởi một số **biểu đồ** (mô hình)
- Một biểu đồ cũng có thể thuộc vào nhiều góc nhìn khác nhau



65

65

Các góc nhìn của UML trong OOAD

- Góc nhìn ca sử dụng (Use case view)
  - là góc nhìn từ ngoài vào HT
  - là cách nhìn của người dùng cuối, người phân tích, người kiểm định
  - không phản ánh tổ chức bên trong mà chỉ làm rõ các chức năng lớn HT phải đáp ứng cho người dùng
  - Sắc thái tĩnh: biểu đồ ca sử dụng
  - Sắc thái động: biểu đồ tương tác, máy trạng thái, biểu đồ hoạt động

66

66

Các góc nhìn của UML trong OOAD

- Góc nhìn thiết kế (Logical view)
  - còn gọi là góc nhìn logic
  - là góc nhìn vào bên trong HT, cho thấy các nhiệm vụ của HT
  - là cách nhìn của người thiết kế HT
  - Sắc thái tĩnh: biểu đồ lớp, biểu đồ đối tượng
  - Sắc thái động: biểu đồ tương tác, máy trạng thái, hoạt động

Các góc nhìn của UML trong OOAD

- Góc nhìn quá trình (Proccess view)
  - còn gọi là góc nhìn song hành
  - phản ánh các lộ trình điều khiển, các quá trình thực hiện, cho thấy sự hoạt động đồng bộ của HT
  - được thể hiện cùng với các biểu đồ như góc nhìn thiết kế, tập trung vào các lớp chủ động
  - Sắc thái động: Biểu đồ trình tự, biểu đồ hành vi …, biểu đồ trạng thái

Các góc nhìn của UML trong OOAD

- Góc nhìn thực thi (Component View)
  - còn gọi là góc nhìn thành phần
  - là góc nhìn đ/v dạng phát hành của phần mềm
  - cho thấy các thành phần và tệp tương đối độc lập, có thể lắp ráp để HT chạy được
  - Sắc thái tĩnh: biểu đồ thành phần
  - Sắc thái động: biểu đồ tương tác, máy trạng thái, hoạt động

69

69

Các góc nhìn của UML trong OOAD

- Góc nhìn triển khai – bố trí (Deployment View)
  - là góc nhìn về hình trạng của phần cứng mà trên đó HT được triển khai
  - nó chỉ rõ sự phân bố, sắp đặt các phần của HT vật lý trên các đơn vị phần cứng
  - Sắc thái tĩnh: biểu đồ bố trí
  - Sắc thái động: biểu đồ tương tác, máy trạng thái, hoạt động

70

70

1.3 Giới thiệu phương pháp mô hình hóa

1.3.1 Khái niệm mô hình và mô hình hóa

1.3.2 Phương pháp mô hình hóa

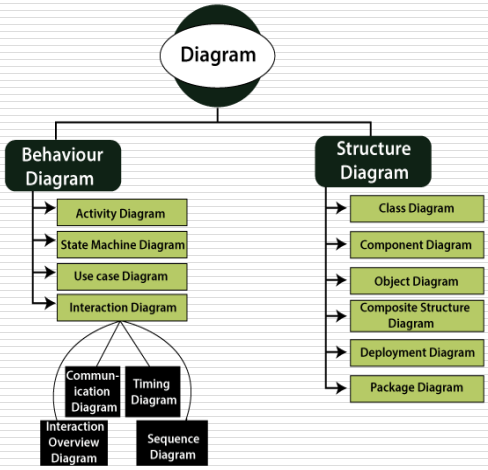
1.3.3 Giới thiệu UML

- 1.3.3.1 UML là gì?
- 1.3.3.2 Lịch sử ra đời và phát triển của UML
- 1.3.3.3 Các miền ứng dụng của UML
- 1.3.3.4 OOAD sử dụng UML như thế nào?
- 1.3.3.5 Các loại biểu đồ trong UML

1.3.3.5 Các loại biểu đồ trong UML

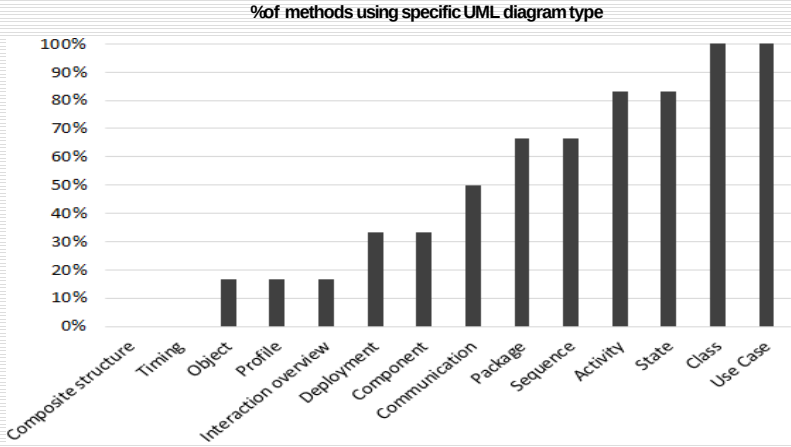
Biểu đồ cấu trúc (Structure Diagram) – Thể hiện cấu trúc tĩnh của hệ thống bằng cách sử dụng các đối tượng, thuộc tính, hoạt động và mối quan hệ

Biểu đồ hành vi (Behaviour Diagram) – Thể hiện hành vi động của hệ thống bằng cách hiển thị sự hợp tác giữa các đối tượng và sự thay đổi trạng thái bên trong của đối tượng



1.3.3.5 Các loại biểu đồ trong UML

UML Diagram Types

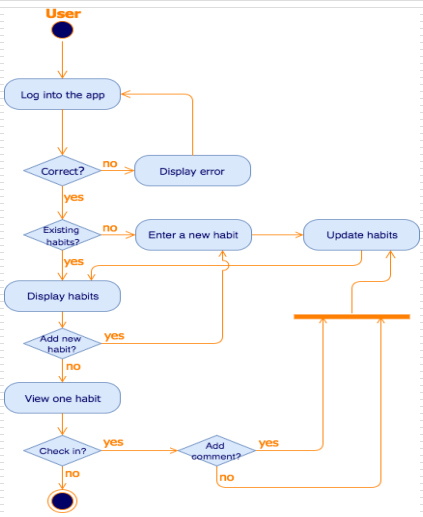


Ref: https://www.researchgate.net/publication/320709806_Topological_UML_Modeling_An_Improved_Approach_for_Domain_Modeling_and_Software_Development

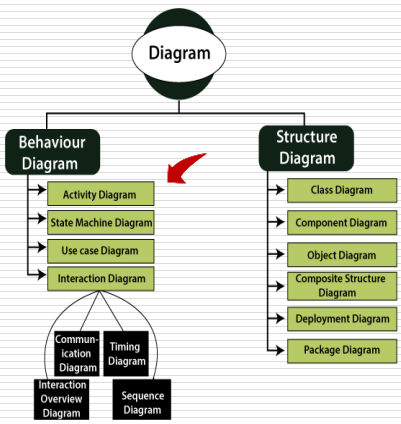
73

Diagram

UML Diagram Types



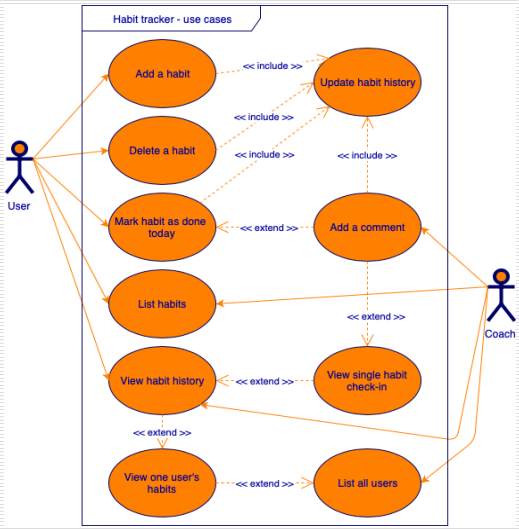
Activity diagram (Biểu đồ hành động)



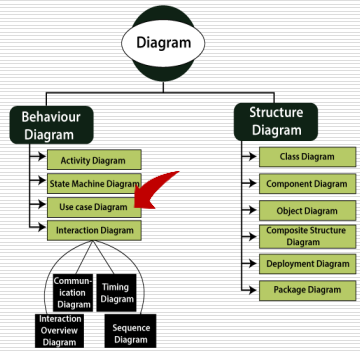
74

Diagram

UML Diagram Types



Use Case Diagram (Biểu đồ ca sử dụng)

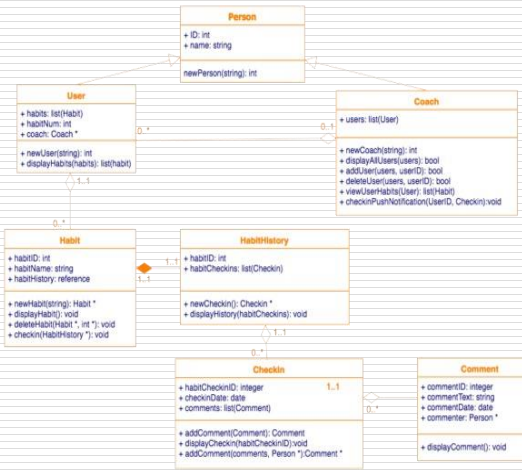


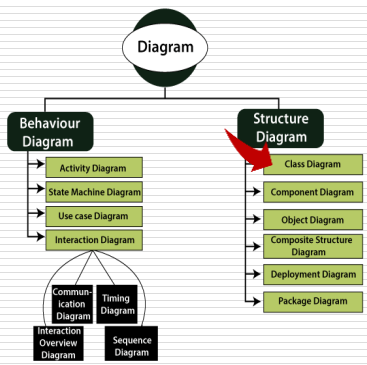
75

Diagram

UML Diagram Types

Class Diagram (Biểu đồ lớp)



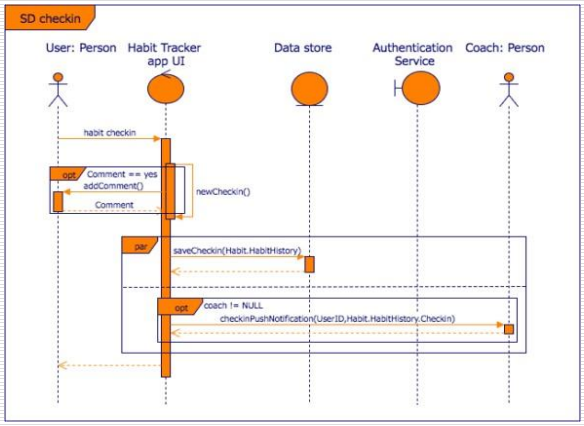


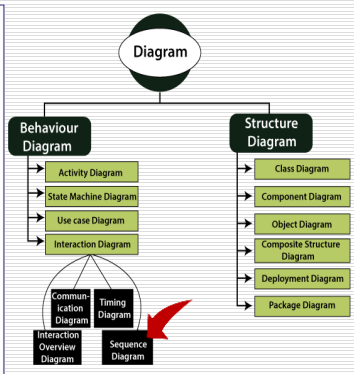
76

Diagram

UML Diagram Types

Sequence Diagram (Biểu đồ tuần tự)



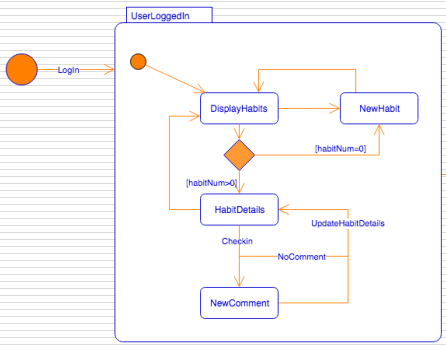


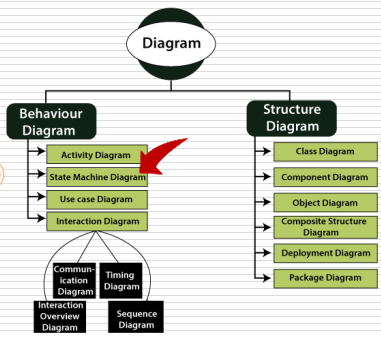
77

Diagram

UML Diagram Types

State Machine Diagram (Biểu đồ trạng thái)



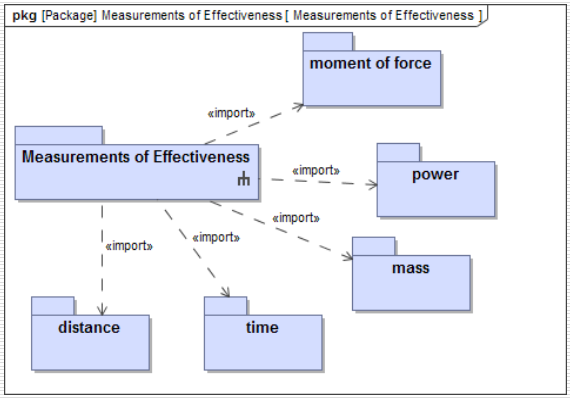


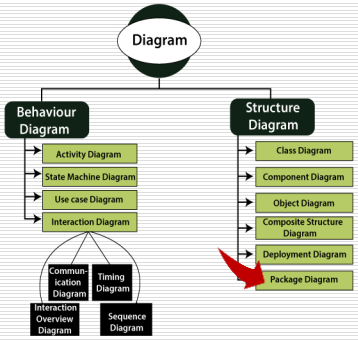
78

Diagram

UML Diagram Types

Package Diagram (Biểu đồ gói)



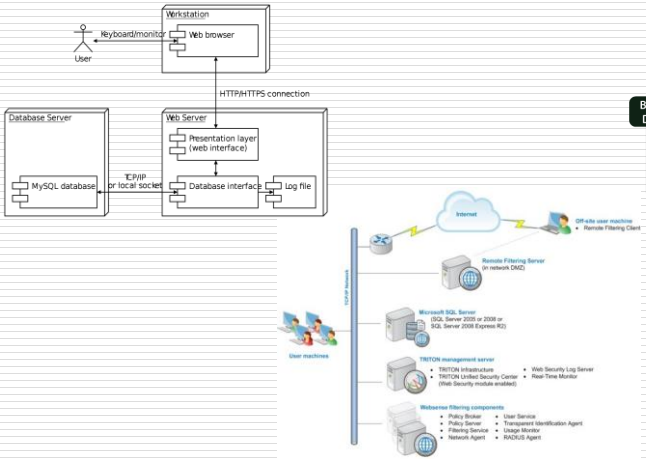


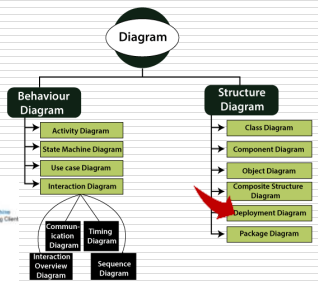
79

Diagram

UML Diagram Types

Deployment Diagram (Biểu đồ triển khai)





80

Các công cụ vẽ biểu đồ UML

- Công cụ mã nguồn mở:
  - EclipseUML
  - UmlDesigner
  - StarUML
  - Argo UML...
- Công cụ thương mại:
  - Enterprise Architect
  - IBM Rational Software Architect
  - Microsoft Visio
  - Visual Paradigm for UML
  - SmartDraw...

Hỏi - đáp

Lời hay ý đẹp

"Có ba thứ ngu dốt: không hiểu biết những gì mình đáng phải biết, hiểu biết không rành những gì mình biết, và hiểu biết những gì mình không cần biết"

LA ROCHEFOUCAULT

83