HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Hoạch định về thiết kế dự án

Giảng viên: Bùi Trọng Đức
Trung tâm nghiên cứu quốc tế về Trí tuệ nhân tạo
Trường Công nghệ Thông tin và Truyền thông

ONE LOVE. ONE FUTURE.

1. Giới thiệu khóa học

- Mục tiêu khoá Level 3:
  - Trang bị kiến thức nền tảng và kỹ năng chuyên sâu về CNTT
  - Xây dựng và quản lý phần mềm
  - Khả năng thích ứng ngay với công việc thực tế dựa trên trình độ hiểu biết về IT tương đương cấp 3 của TOPCIT: mức thành tạo, đủ năng lực vận dụng công nghệ trong môi trường làm việc thực tế
- TOPCIT: **Test of Practical Competency in ICT** (**Bài kiểm tra năng lực thực tiễn trong công nghệ thông tin và truyền thông**).
  - Hệ thống đánh giá năng lực được phát triển bởi **Viện Phát triển Xã hội Thông tin Hàn Quốc (NIA)**
  - Năng lực CNTT và khả năng ứng dụng CNTT vào công việc thực tiễn, đặc biệt trong môi trường doanh nghiệp.

1. Giới thiệu khóa học

- **Mục đích của TOPCIT:**
  - Đánh giá năng lực toàn diện của sinh viên hoặc nhân lực CNTT trong các lĩnh vực **phát triển phần mềm**, **cơ sở dữ liệu**, **mạng máy tính**, **an toàn thông tin**, **giao tiếp doanh nghiệp**, v.v.
  - Giúp **các doanh nghiệp** tuyển dụng nhân sự có kỹ năng CNTT phù hợp với nhu cầu thực tiễn.
  - Hỗ trợ **trường đại học** trong việc cải tiến chương trình đào tạo, gắn kết đào tạo với nhu cầu thực tế.

1. Giới thiệu khóa học

- **Cấu trúc bài thi TOPCIT: 4 lĩnh vực chính**, chia thành hai nhóm:

- **Năng lực công nghệ thông tin (Technical Competency):**
  - Phát triển phần mềm (Software Development)
  - Cơ sở dữ liệu (Database)
  - Mạng & Bảo mật (Network & Security)

- **Năng lực kinh doanh và giao tiếp (Business & Communication):**
  - Hiểu biết về kinh doanh (Business Understanding)
  - Kỹ năng giao tiếp (Communication Skills)

1. Giới thiệu khoá học

- **Cấu trúc bài thi TOPCIT:**

Loại câu hỏi	Tỷ lệ điểm (%)	Nội dung đánh giá chính
Trắc nghiệm	19%	Hiểu biết về các khái niệm và thuật ngữ cốt lõi
Điền từ	12%	Khả năng đưa ra thuật ngữ chính xác, kết quả tính toán
Tự luận	21%	Khả năng mô tả hợp lý nguyên nhân, giải pháp cho một tình huống cụ thể
Thực hành	48%	Viết mã nguồn, chạy truy vấn SQL, thiết kế sơ đồ, v.v.

1. Giới thiệu khoá học

- **Cấu trúc các lĩnh vực đánh giá TOPCIT:**

Phân loại	Lĩnh vực chi tiết	Số câu	Điểm
Lĩnh vực công nghệ (565 điểm)	Phát triển phần mềm	17	260
	Hiểu và sử dụng dữ liệu	9	145
	Kiến trúc hệ thống	8	75
	Bảo mật thông tin	8	85
	Kinh doanh IT và đạo đức	11	95
Lĩnh vực kinh doanh (175 điểm)	Giao tiếp kỹ thuật / Quản lý dự án	9	80
Lĩnh vực tích hợp (260 điểm)	Giải quyết vấn đề tích hợp Công nghệ & Kinh doanh	3	260

1. Giới thiệu khoá học

- **Hệ thống level TOPCIT:**

Level	Tên gọi	Khoảng điểm	Định nghĩa chi tiết
Cấp 1	Mức độ mới bắt đầu	0 ~ 149	Mức độ hiểu biết về kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh còn hạn chế, cần học hỏi thêm
Cấp 2	Mức độ thử thách	150 ~ 399	Mức độ hiểu biết kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh
Cấp 3	Mức độ thành thạo	400 ~ 649	Mức độ có thể áp dụng kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh để giải quyết các vấn đề
Cấp 4	Mức độ giải quyết vấn đề	650 ~ 849	Mức độ có thể ứng dụng kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh để giải quyết các vấn đề phức tạp hơn
Cấp 5	Mức độ sáng tạo và tổng hợp	850 ~ 1,000	Mức độ có thể ứng dụng kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh dựa trên sự sáng tạo để đưa ra các giải pháp mới và chủ động giải quyết các vấn đề

1. Giới thiệu khoá học

- **Level 2 TOPCIT:** có kiến thức và kỹ năng cơ bản trong lĩnh vực công nghệ và kinh doanh SW/IT, nhưng cần được đào tạo thêm

Lĩnh vực	Năng lực cốt lõi yêu cầu
Phát triển phần mềm	- Hiểu các giai đoạn của vòng đời phát triển phần mềm (SDLC) và có thể giải thích các khái niệm về các phương pháp phát triển điển hình (thác nước, agile, v.v.). - Hiểu các cấu trúc dữ liệu cơ bản như mảng, ngăn xếp, hàng đợi và các khái niệm thuật toán cơ bản như sắp xếp, tìm kiếm. - Nắm được các khái niệm cơ bản về lập trình hướng đối tượng (lớp, đối tượng, kế thừa, v.v.).
Hiểu và sử dụng dữ liệu	- Hiểu các thuật ngữ cơ sở dữ liệu cơ bản như cơ sở dữ liệu, bảng, bản ghi. - Biết được sự cần thiết của mô hình hóa dữ liệu và các khái niệm cơ bản về chuẩn hóa. - Hiểu cú pháp cơ bản của các câu lệnh SQL đơn giản.
Kiến trúc hệ thống	- Phân biệt các thành phần cơ bản của hệ thống IT như máy chủ, máy khách, mạng, hệ điều hành và giải thích vai trò của từng thành phần. - Hiểu ý nghĩa của các thuật ngữ mạng cơ bản như địa chỉ IP, cổng.

1. Giới thiệu khoá học

- **Level 2 TOPCIT:** có kiến thức và kỹ năng cơ bản trong lĩnh vực công nghệ và kinh doanh SW/IT, nhưng cần được đào tạo thêm

Lĩnh vực	Năng lực cốt lõi yêu cầu
Bảo mật thông tin	- Hiểu sự cần thiết của bảo mật thông tin và các khái niệm về bảo mật, tính toàn vẹn, tính sẵn sàng. - Biết các loại và đặc điểm của các mối đe dọa bảo mật cơ bản như virus, phần mềm độc hại, lừa đảo.
Kinh doanh IT và đạo đức nghề nghiệp	- Biết về các công ty và dịch vụ IT tiêu biểu, và có kiến thức cơ bản về xu hướng mới nhất trong ngành IT. - Nhận thức được tầm quan trọng của đạo đức bảo mật thông tin và bảo vệ thông tin cá nhân
Giao tiếp kỹ thuật và quản lý dự án	- Hiểu mục đích và định dạng của các tài liệu kỹ thuật cơ bản như email công việc, báo cáo đơn giản. - Biết ý nghĩa của các thuật ngữ cơ bản trong quản lý dự án như dự án, lịch trình, tài nguyên.

1. Giới thiệu khoá học

- **Level 3 TOPCIT:** có thể áp dụng kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh để giải quyết các vấn đề

Lĩnh vực	Năng lực cốt lõi yêu cầu
Phát triển phần mềm	- Hiểu tài liệu đặc tả yêu cầu và có thể sử dụng các công cụ thiết kế như UML để thiết kế cấu trúc phần mềm. - Áp dụng các nguyên tắc lập trình hướng đối tượng để triển khai các chức năng cơ bản và viết mã ổn định bao gồm xử lý ngoại lệ. - Phát hiện các vấn đề đơn giản trong mã (Code Smell) và cải thiện chúng thông qua tái cấu trúc.
Hiểu và sử dụng dữ liệu	- Thiết kế mô hình dữ liệu logic/vật lý phù hợp với các yêu cầu đã cho và thực hiện quá trình chuẩn hóa. - Sử dụng SQL cơ bản (SELECT, JOIN, v.v.) để trích xuất và thao tác dữ liệu cần thiết. - Hiểu các khái niệm cơ bản về các công nghệ mới như trí tuệ nhân tạo, dữ liệu lớn và có thể giải thích cách chúng có thể được áp dụng vào các vấn đề kinh doanh đơn giản.
Kiến trúc hệ thống	- Hiểu vai trò của các thành phần hệ thống như máy chủ, máy khách, cơ sở dữ liệu, phần mềm trung gian và có thể cấu hình kiến trúc hệ thống đơn giản. - Hiểu cách hoạt động của các giao thức mạng cơ bản (TCP/IP) và có thể suy luận nguyên nhân của các tình huống lỗi mạng đơn giản.

1. Giới thiệu khoá học

- **Level 3 TOPCIT:** có thể áp dụng kiến thức và kỹ năng trong lĩnh vực công nghệ và kinh doanh để giải quyết các vấn đề

Lĩnh vực	Năng lực cốt lõi yêu cầu
Bảo mật thông tin	- Hiểu 3 yếu tố chính của bảo mật thông tin (tính bảo mật, tính toàn vẹn, tính sẵn sàng) và áp dụng vào các trường hợp thực tế để phân tích các mối đe dọa bảo mật. - Hiểu nguyên lý của các lỗ hổng web điển hình như SQL Injection, XSS và áp dụng các nguyên tắc mã hóa bảo mật cơ bản.
Kinh doanh IT và đạo đức nghề nghiệp	- Phân tích các trường hợp mô hình kinh doanh sử dụng công nghệ IT và đề xuất mô hình doanh thu cho một công nghệ cụ thể. - Đưa ra phán đoán phù hợp trong các tình huống khó xử về đạo đức và pháp luật liên quan đến IT như luật bảo vệ thông tin cá nhân, bản quyền.
Giao tiếp kỹ thuật và quản lý dự án	- Viết các tài liệu kỹ thuật có cấu trúc hợp lý phù hợp với mục đích như kế hoạch dự án, báo cáo kết quả. - Hiểu WBS (Cấu trúc phân chia công việc) và có thể lập kế hoạch phạm vi và lịch trình cho các dự án nhỏ.

1. Giới thiệu khoá học

- Nguyên lý lập trình hướng đối tượng OOP
- Các mô hình phát triển phần mềm
- Nguyên lý xây dựng và phát triển phần mềm: SOLID, ...
- Quản lý dự án phần mềm: Github, Notion, Google Docs
- Phân tích và thiết kế phần mềm
- Các loại biểu đồ UML: Biểu đồ ca sử dụng, biểu đồ class, biểu đồ trình tự, biểu đồ giao tiếp, ...

Các nguyên lý

- Biểu đồ ERD
- Thiết kế cơ sở dữ liệu
- Truy vấn dữ liệu
- Các thao tác truy vấn nâng cao

Cơ sở dữ liệu

- Khái niệm REST API
- Cài đặt và khởi chạy dự án Spring Boot
- Mô hình nhiều tầng của Spring Boot
- Kết nối API

- Kết nối DB với Spring Boot
- Các thao tác xử lý DB thông qua Spring Boot

API và Spring Boot

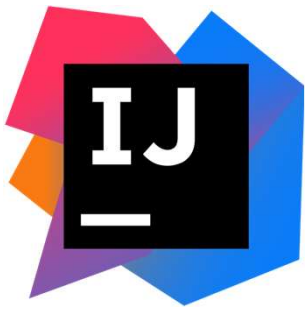
- Tổng quan về xác thực và JWT: Giới thiệu về xác thực và phân quyền, JWT
- Xây dựng API cơ bản: đăng ký/đăng nhập, mã hóa mật khẩu
- Triển khai JWT Authentication: Sinh JWT khi đăng nhập thành công, Gửi token về client, Filter xác thực, bảo vệ API

Xác thực và phân quyền

- Xử lý ngoại lệ
- Các mô hình kiểm thử
- Kiểm thử API

Kiểm thử

1. Giới thiệu khoá học



POSTMAN

1. Giới thiệu khoá học

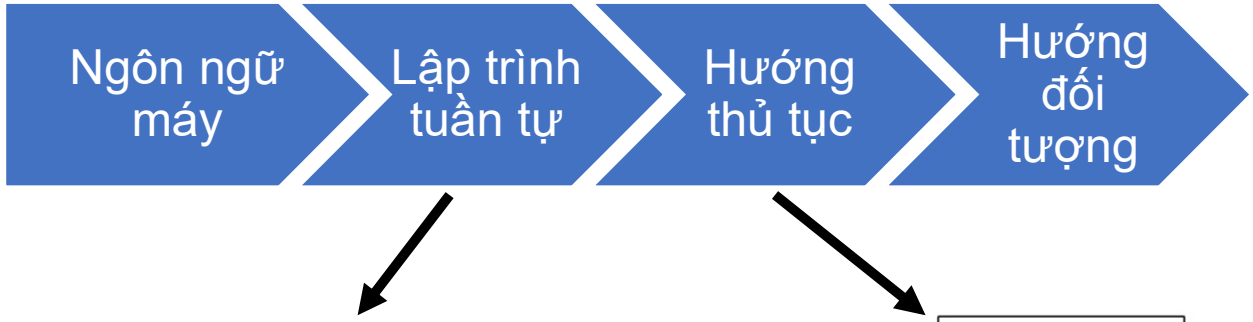
- **Tài liệu**
  - Giáo trình bài giảng slide
  - TOPCIT Essence Ver. 3 Technical Field:
    - 01 Software Development
    - 02 Understanding and Using Data
    - 03 Overview of System Architecture
    - 04 Understanding Information Security
  - Link: https://www.topcit.or.kr/home.do

Kỹ thuật lập trình

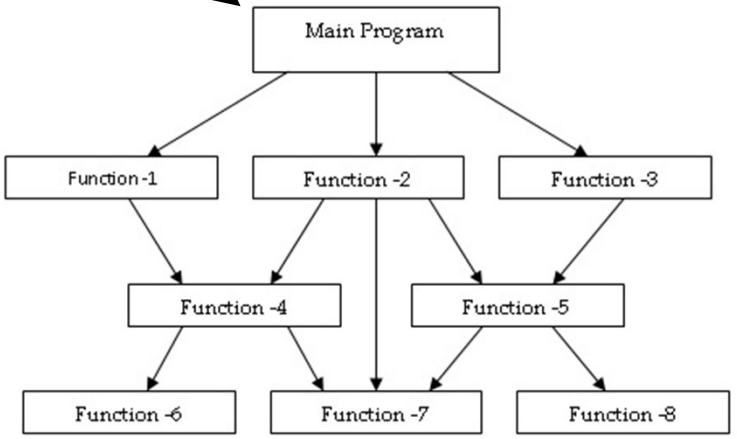
- Kỹ thuật lập trình: Kỹ thuật thực thi một giải pháp phần mềm (cấu trúc dữ liệu + giải thuật) dựa trên nền tảng một phương pháp luận (methodology) và một hoặc nhiều ngôn ngữ lập trình phù hợp với yêu cầu đặc thù của ứng dụng
- Ngôn ngữ lập trình
  - Là ngôn ngữ được chuẩn hóa
  - Cả con người và máy tính có thể đọc và hiểu được
  - Sử dụng chương trình dịch tương ứng để biên dịch toàn bộ chương trình nguồn thành mã máy trước khi thực hiện

Ngôn ngữ lập trình

- Sự phát triển của ngôn ngữ lập trình



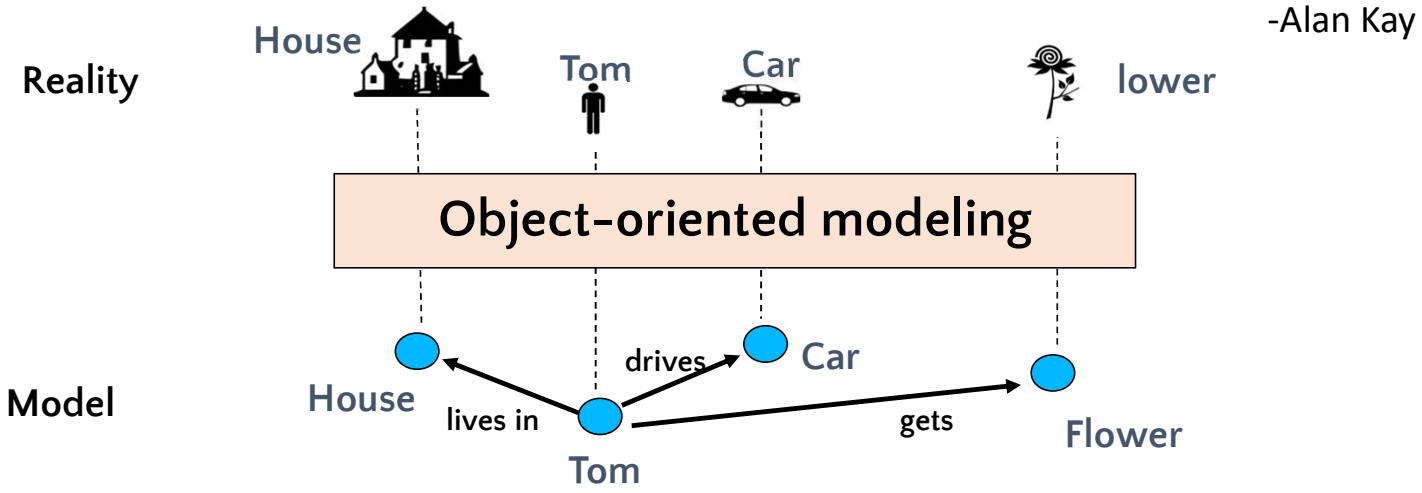

```
        RELOAD EQU 0E6H    ; defining reload constant for baudrate g
        ORG 0000H          ; org directive
        SJMP START         ; jump to main program

SENDCH: ORG 0023H          ; ISR for RI/TI interrupt
        CLR TI             ; clear transmit flag
        MOV SBUF, #'A'     ; send the ASCII value of 'A'
        RETI               ; return back to main program

START:  ANL PCON, #7FH     ; Set SMOD=0
        ANL TMOD, #0FH     ; Alter only the setting of Timer-1
        ORL TMOD, #20H     ; Timer-1 in mode-2
        MOV TH1, #RELOAD   ; Move the reload value to TH1
        SETB TR1           ; start Timer-1 for baud rate generation
        MOV SCON, #40H     ; set serial port in mode-1
        ORL IE, #90H       ; Enable serial port interrupt

        MOV SBUF, #'A'     ; Transmit a character

WAIT:   SJMP WAIT          ; wait till interrupt occurs

        END
```



Hướng đối tượng

- Thể hiện các thành phần của bài toán là các "đối tượng" (object).
- Hướng đối tượng là một kỹ thuật để mô hình hóa hệ thống thành nhiều đối tượng tương tác với nhau

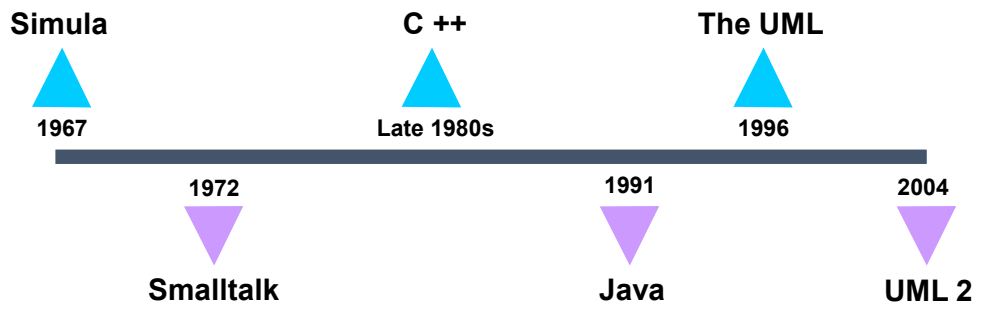
"Tất cả mọi thứ đều là đối tượng"
-Alan Kay



Tổng quan về lập trình hướng đối tượng

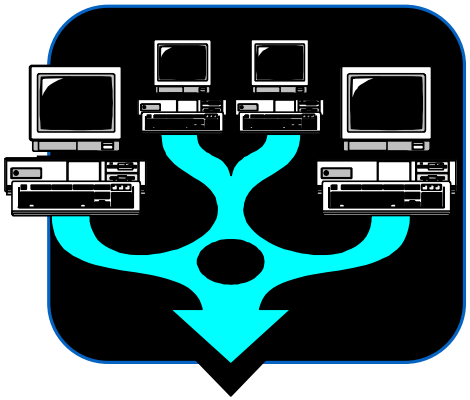
- Công nghệ đối tượng là một tập các quy tắc (trừu tượng hóa, đóng gói, đa hình), các hướng dẫn để xây dựng phần mềm, cùng với ngôn ngữ, cơ sở dữ liệu và các công cụ khác hỗ trợ các quy tắc này.
- Các mốc chính của công nghệ đối tượng

(*Object Technology - A Manager's Guide*, Taylor, 1997)



Tổng quan về lập trình hướng đối tượng

- Các hệ thống Client/Server và phát triển Web
- Hệ nhúng (embedded system)
- Hệ thống thời gian thực (real-time)
- Hệ thống phần mềm nói chung…



Tổng quan về lập trình hướng đối tượng

- Đối tượng là gì?
- Đối tượng trong thế giới thực, là một thực thể cụ thể mà thông thường chúng ta có thể *sờ, nhìn thấy* hay *cảm nhận* được.
- Tất cả có trạng thái (state) và hành vi (behaviour)

	Trạng thái	Hành động	
Con chó	Tên Màu Giống Vui sướng	Sủa Vẫy tai Chạy Ăn	
Xe đạp	Bánh răng Bàn đạp Dây xích Bánh xe	Tăng tốc Giảm tốc Chuyển bánh răng ...	

Trạng thái và hành vi

- Trạng thái của một đối tượng là một trong các điều kiện tại đó mà đối tượng tồn tại
- Trạng thái của một đối tượng có thể thay đổi theo thời gian

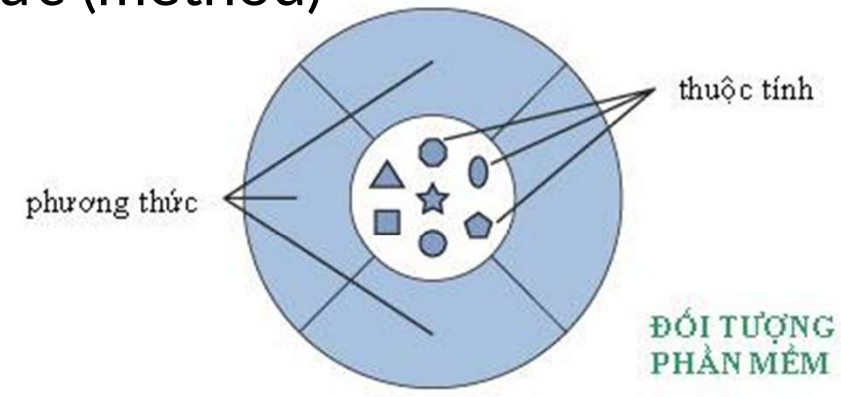
Name: J Clark
Employee ID: 567138
Date Hired: July 25, 1991
Status: Tenured
Discipline: Finance
Maximum Course Load: 3 classes

- Hành vi quyết định đối tượng đó hành động và đáp trả như thế nào đối với bên ngoài
- Hành vi nhìn thấy được của một đối tượng được mô hình thành một tập các thông điệp nó có thể đáp trả (các thao tác mà đối tượng đó thực hiện)

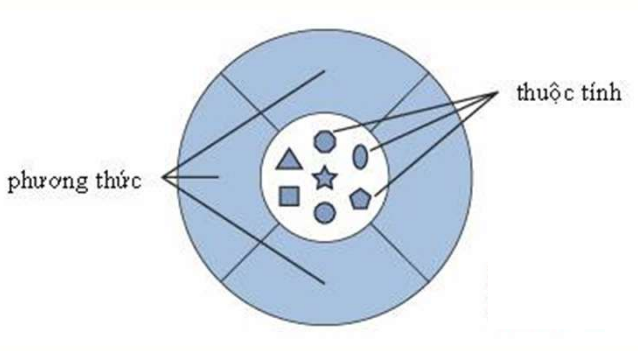
Professor Clark's behavior
Submit Final Grades
Accept Course Offering
Take Sabbatical

Đối tượng phần mềm

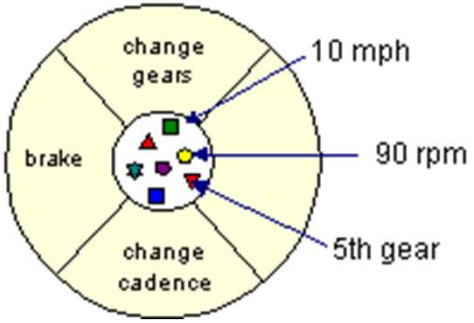
- Các đối tượng phần mềm được dùng để biểu diễn các đối tượng thế giới thực.
- Cũng có trạng thái và hành vi
- Trạng thái: thuộc tính (attribute; property)
- Hành vi: phương thức (method)



Ví dụ đối tượng phần mềm



Đối tượng phần mềm



Đối tượng phần mềm **Xe Đạp**

Đối tượng (object) là một thực thể phần mềm bao bọc các **thuộc tính** và các **phương thức** liên quan.

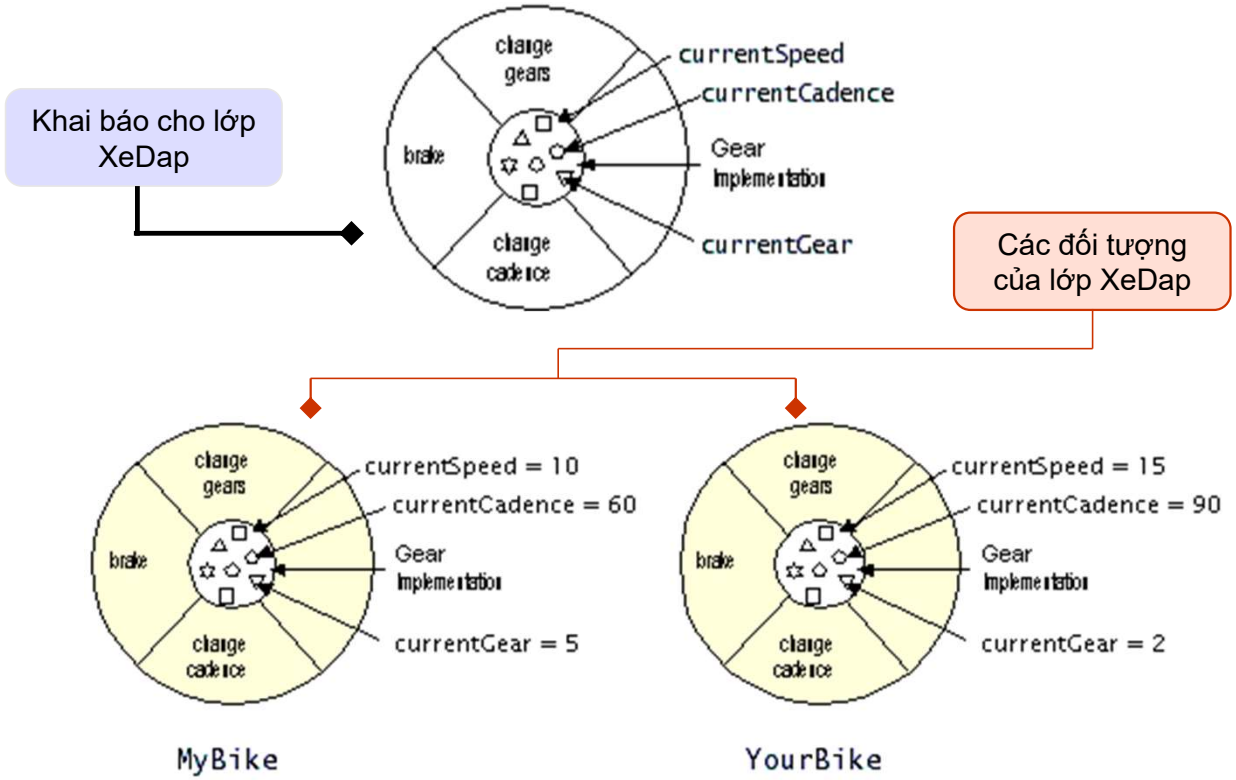
Thuộc tính được xác định bởi giá trị cụ thể gọi là **thuộc tính thể hiện**.

Một đối tượng cụ thể được gọi là một **thể hiện**.

Lớp và đối tượng

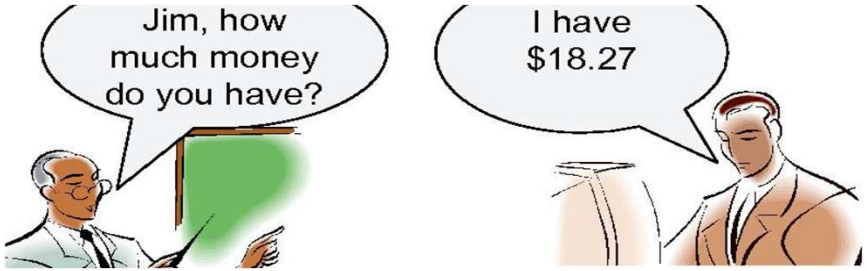
- Một **lớp** là một thiết kế (blueprint) hay mẫu (prototype) cho các đối tượng cùng kiểu
  - Ví dụ: lớp XeDap là một thiết kế chung cho nhiều đối tượng xe đạp được tạo ra
- Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó
- Một **đối tượng** là một thể hiện cụ thể của một lớp.
  - Ví dụ: mỗi đối tượng xe đạp là một thể hiện của lớp XeDap
- Mỗi thể hiện có thể có những thuộc tính thể hiện khác nhau
  - Ví dụ: một xe đạp có thể đang ở số (gear) thứ 5 trong khi một xe khác có thể là đang ở số (gear) thứ 3.

Ví dụ lớp Bike



Tương tác giữa các đối tượng

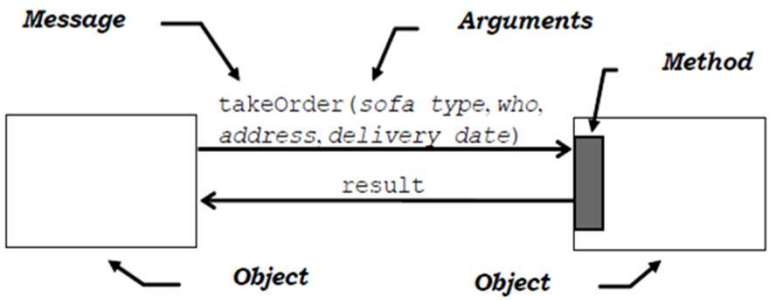
- Sự giao tiếp giữa các đối tượng trong thế giới thực:



- Các đối tượng và sự tương tác giữa chúng trong lập trình
  - Các đối tượng giao tiếp với nhau bằng cách gửi thông điệp (message)

Thông điệp vs. Phương thức

- Thông điệp
  - Được gửi từ đối tượng này đến đối tượng kia, không bao gồm đoạn mã thực sự sẽ được thực thi.
- Phương thức
  - Thủ tục/hàm trong ngôn ngữ lập trình cấu trúc.
  - Là sự thực thi dịch vụ được yêu cầu bởi thông điệp.
  - Là đoạn mã sẽ được thực thi để đáp ứng thông điệp được gửi đến cho đối tượng.



Hướng cấu trúc vs. Hướng đối tượng

- Hướng cấu trúc:
  - data structures + algorithms = Program
  - (cấu trúc dữ liệu + giải thuật = Chương trình)
- Hướng đối tượng:
  - objects + messages = Program
  - (đối tượng + thông điệp = Chương trình)

Các nguyên lý cơ bản của OOP

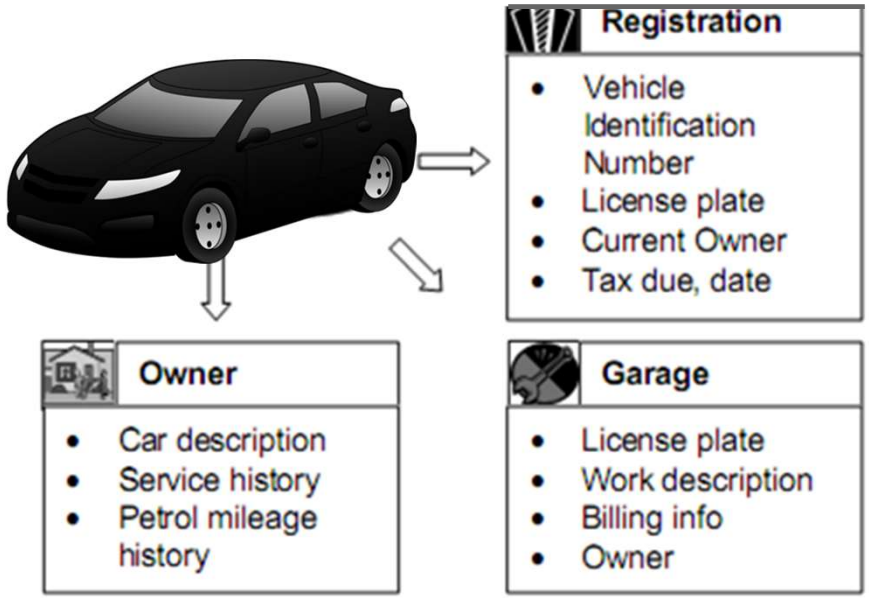
- Trừu tượng hóa (Abstraction)
- Đóng gói (Encapsulation)
- Kế thừa (Inheritance)
- Đa hình (Polymorphism)

Trừu tượng hóa

- Là quá trình loại bỏ đi các thông tin/tính chất cụ thể và giữ lại những thông tin/tính chất chung.
- Tập trung vào các đặc điểm cơ bản của thực thể, các đặc điểm phân biệt nó với các loại thực thể khác.
- Phụ thuộc vào góc nhìn (Quan trọng trong ngữ cảnh này nhưng lại không có ý nghĩa nhiều trong ngữ cảnh khác)

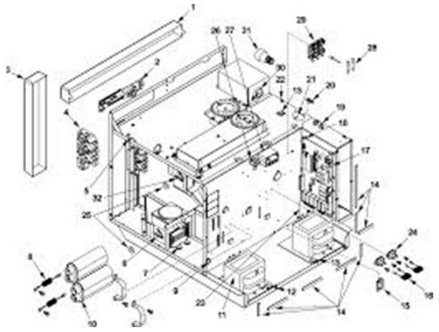
Trừu tượng hóa

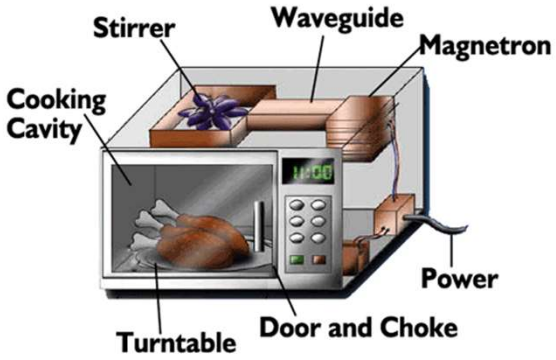
Phụ thuộc vào góc nhìn



Đóng gói

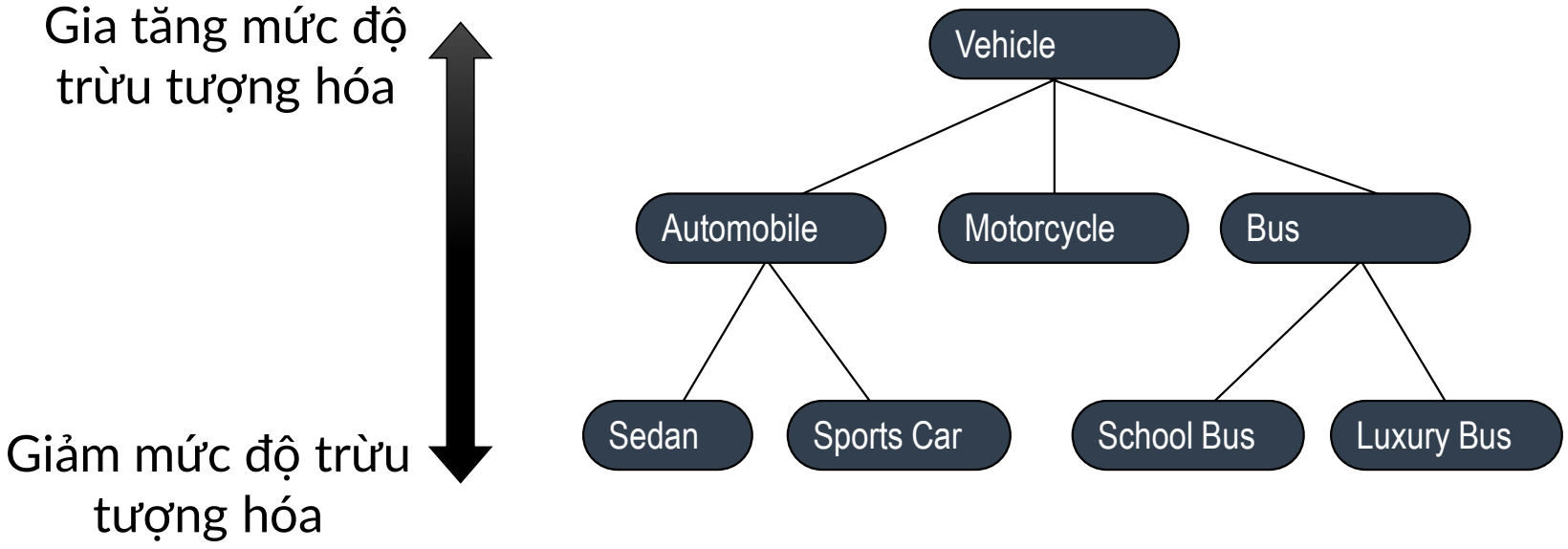
- Che giấu, ẩn đi chi tiết thực hiện bên trong
- Cung cấp cho thế giới bên ngoài một giao diện
- Việc sử dụng không ảnh hưởng bởi chi tiết bên trong.





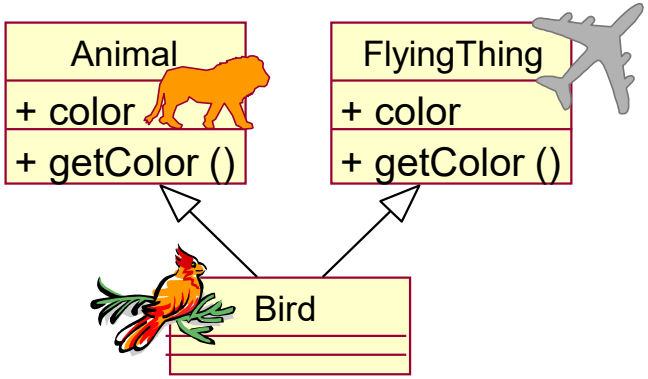
Kế thừa

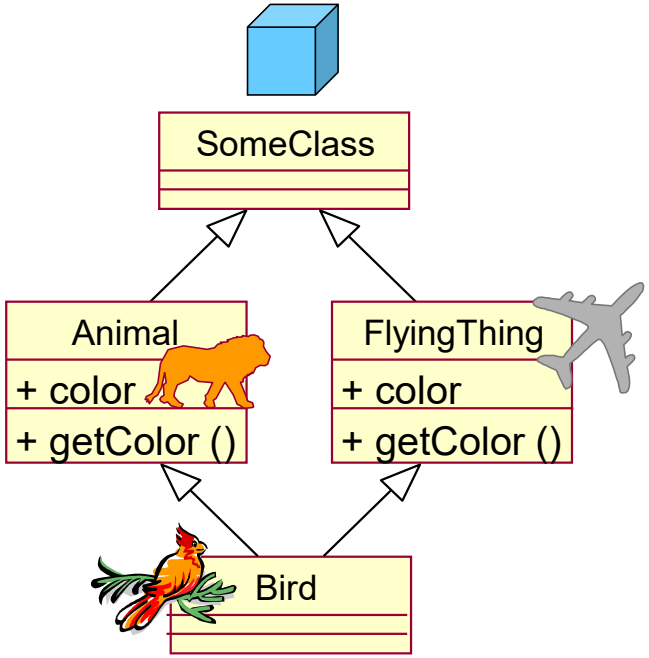
- Xếp hạng hay xếp thứ tự các mức trừu tượng vào một cấu trúc cây
- Các phần tử ở cùng cấp trong sơ đồ phân cấp thì có cùng mức trừu tượng hóa



Kế thừa

- Vấn đề đa kế thừa trong Java
  - Name collision
  - "Diamond shape" problem





Đa hình

- Đa hình: “one name, many forms”
- Nạp chồng phương thức: phương thức cùng tên, nhưng hoạt động khác nhau
- Add(int x, int y)
- Add(float x, float y)
- Add(float x, float y, float z)
- Ghi đè phương thức (Method Overriding)
- Một Intern (thực tập sinh) là một Intern, đồng thời cũng có thể được xem là một Staff (nhân viên)
- Phương thức quẹt thẻ của Intern khác với phương thức quẹt thẻ của Staff

Vòng đời phần mềm

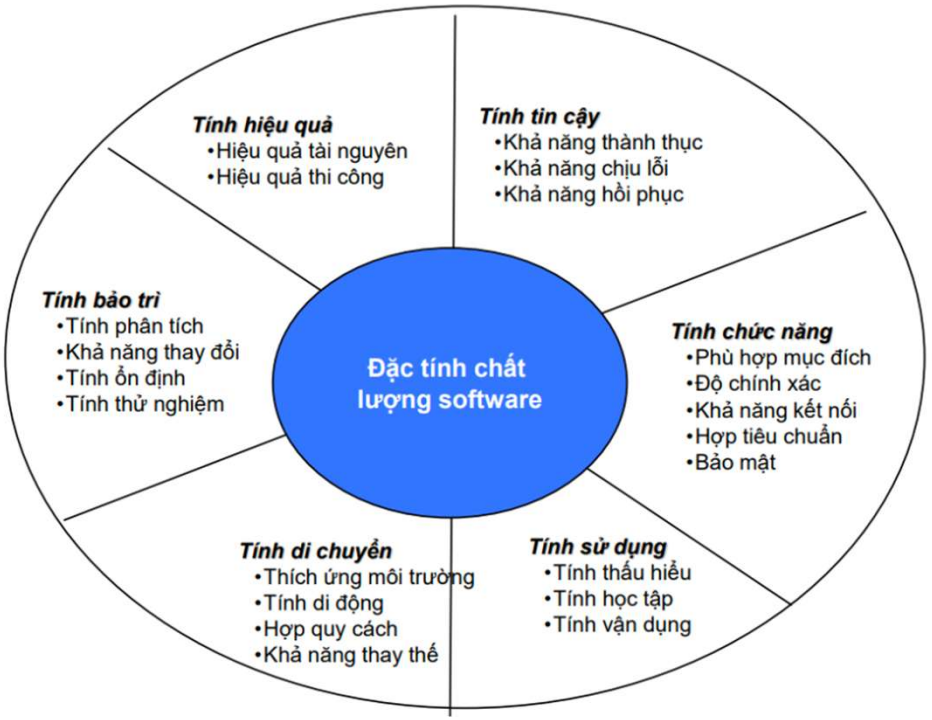
- Tên tiếng Anh: Software life cycle
- Vòng đời phần mềm là thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không đâu dùng)
- Quy trình phần mềm (vòng đời phần mềm) được phân chia thành các pha chính: phân tích, thiết kế, chế tạo, kiểm thử, bảo trì. Biểu diễn các pha có thể khác nhau theo từng mô hình

Vòng đời phát triển phần mềm

- Tên tiếng Anh: Software Development Life Cycle- SDLC
- Vòng đời phát triển phần mềm (SDLC) là quy trình về chi phí và thời gian mà các nhóm phát triển sử dụng để thiết kế và xây dựng phần mềm.
- Mục tiêu của SDLC là giảm thiểu rủi ro dự án thông qua việc lập kế hoạch trước để phần mềm đáp ứng mong đợi của khách hàng trong giai đoạn sản xuất và hơn thế nữa.
- Phương pháp này vạch ra một loạt các bước chia quy trình phát triển phần mềm thành các nhiệm vụ mà có thể chỉ định, hoàn thành và đo lường.

Quy trình phát triển PM

- Mục tiêu: Đảm bảo rằng sản phẩm cuối cùng đáp ứng các yêu cầu của khách hàng và đạt được các tiêu chuẩn chất lượng mong muốn.
- Quy trình tốt:
  - dẫn tới phần mềm tốt
  - giảm thiểu rủi ro
  - rõ ràng, dự án dễ quản lý
  - cho phép làm việc nhóm



Quy trình phát triển PM

- Kiểm soát chất lượng PM
  - Xác thực các yêu cầu.
  - Xác nhận hệ thống và thiết kế chương trình.
  - Kiểm tra khả năng sử dụng.
  - Kiểm tra chương trình.
  - Kiểm tra chấp nhận.
  - Sửa lỗi và bảo trì.

Quy trình phát triển PM

- **Các bước cơ bản trong phát triển phần mềm**
  - Thu thập yêu cầu (Requirements Gathering/Feasibility Study)
  - Phân tích yêu cầu (Requirement Analysis)
  - Thiết kế (Design)
  - Phát triển (Development)
  - Kiểm thử (Testing)
  - Triển khai (Deployment)
  - Bảo trì (Maintenance)

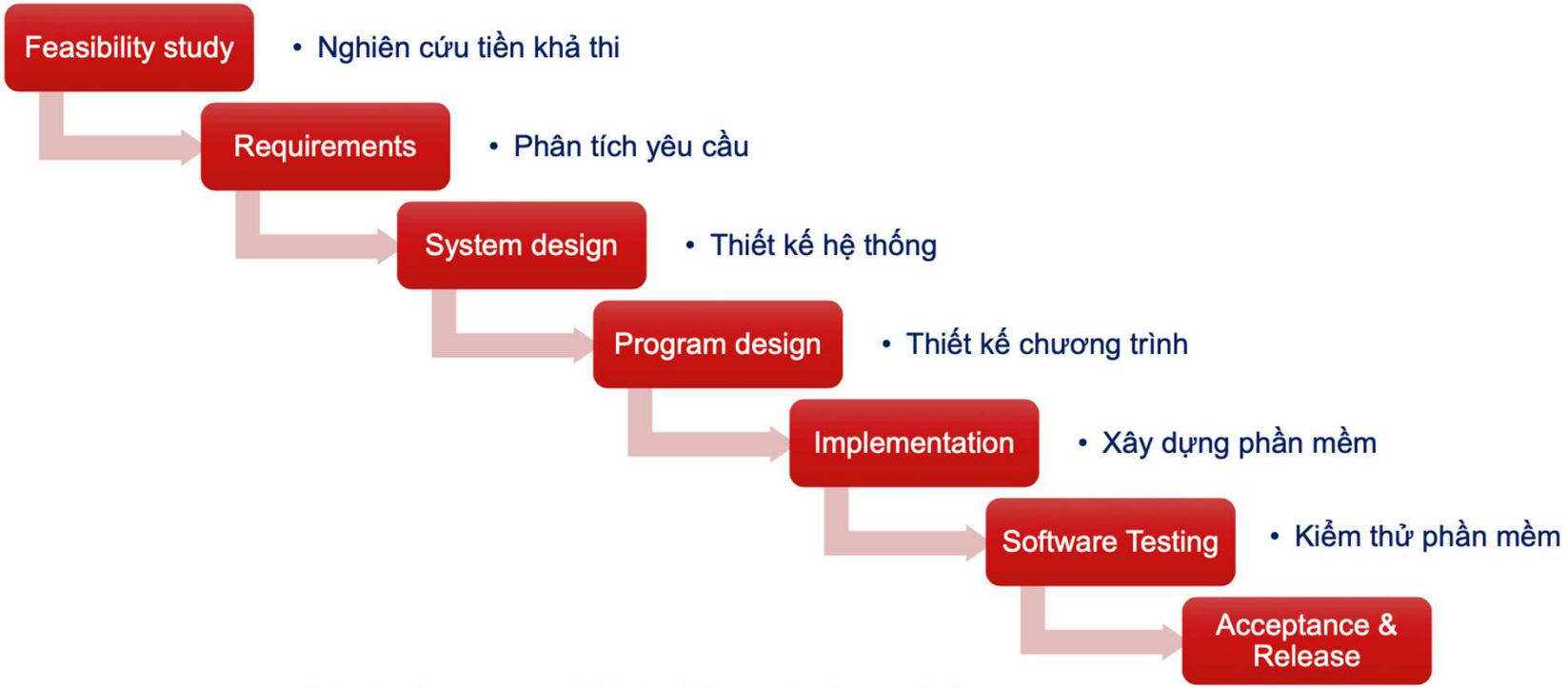
Có thể lặp đi lặp lại trên thực tế

Quy trình phát triển PM

- **Các quy trình trong thực tế**
  - Quy trình phát triển phần mềm trong thực tế thường được thiết lập dựa trên các mô hình hoặc phương pháp phát triển phần mềm đã được kiểm chứng và sử dụng rộng rãi.
  - Một số quy trình phổ biến:
    - mô hình Waterfall (thác nước)
    - mô hình Agile
    - mô hình Incremental (Tăng trưởng)
    - mô hình Spiral (xoáy ốc)
    - mô hình V, DevOps
    - mô hình triển khai và tích hợp liên tục (Continuous Integration/Continuous Deployment - CI/CD)
  - Các quy trình này thường được tùy chỉnh và điều chỉnh để phù hợp với yêu cầu cụ thể của dự án và tổ chức. Chúng cũng có thể được kết hợp hoặc mở rộng để tạo ra một quy trình phát triển độc đáo phù hợp với hoàn cảnh cụ thể.
  - Không có quy trình phát triển nào là hoàn hảo và phù hợp với tất cả các tình huống. Việc tùy chỉnh và điều chỉnh quy trình để phù hợp với dự án cụ thể và nhóm phát triển là rất quan trọng.

Quy trình phát triển PM

- **MH1. Mô hình thác nước (Waterfall)**



Các bước trong mô hình thác nước (waterfall)

Quy trình phát triển PM

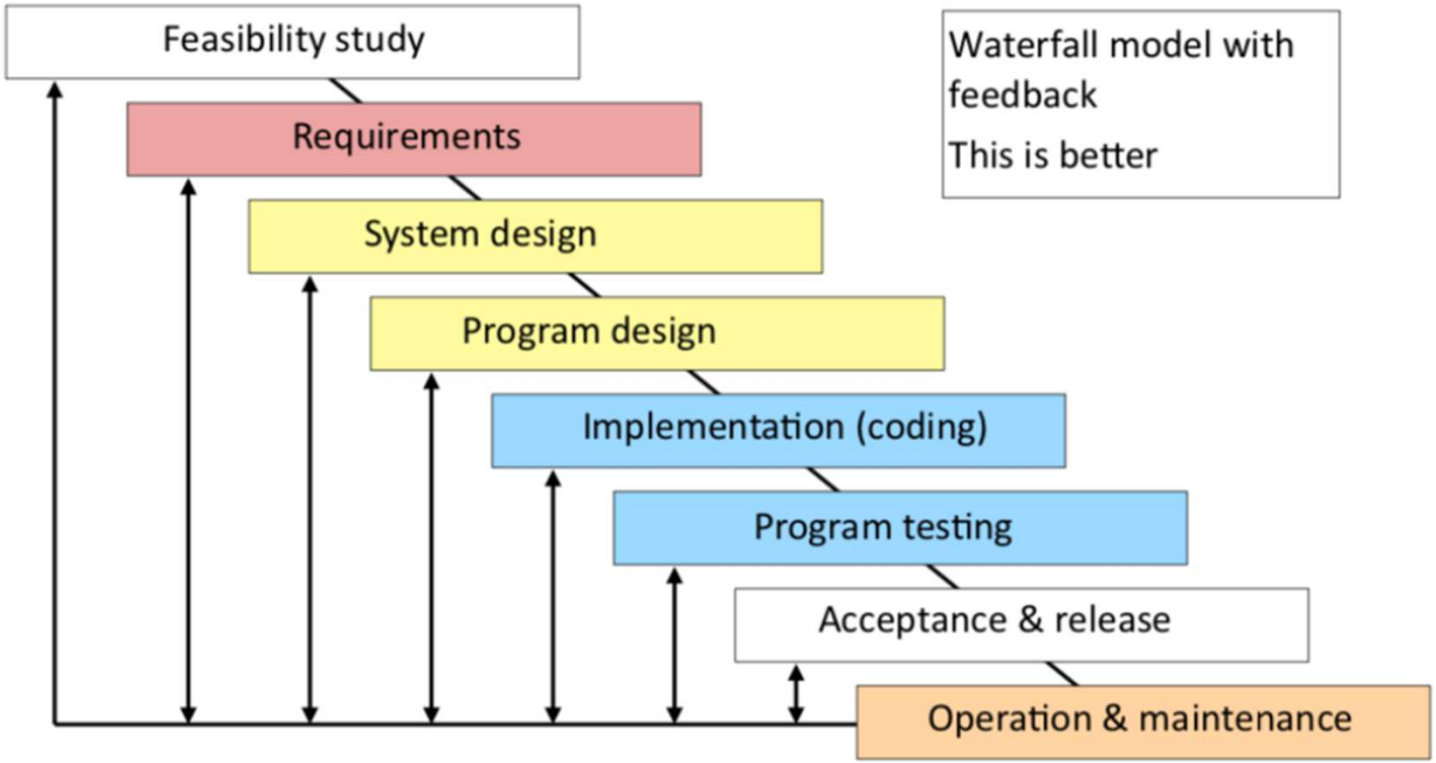
- **Mô hình thác nước (Waterfall)**
  - lâu đời nhất; được đề xuất bởi Winston Royce vào năm 1970.
  - Mô hình này được gọi là thác nước vì nó thường được vẽ với một chuỗi các hoạt động qua các giai đoạn của vòng đời "xuống dốc" từ trái sang phải: phân tích, yêu cầu, đặc tả, thiết kế, cài đặt, kiểm thử, bảo trì
- Có nhiều phiên bản của mô hình thác nước:
  - Các giai đoạn / hoạt động có thể được cấu trúc theo các mức độ chi tiết khác nhau
  - Phản hồi có thể linh hoạt hơn hoặc ít hơn

Quy trình phát triển PM

- Ưu điểm:
  - Tách nhiệm vụ trong mỗi giai đoạn 1 cách riêng biệt
  - Khả năng hiển thị tốt, dễ theo dõi
  - Quy trình kiểm soát chất lượng ở mỗi bước
  - Giám sát chi phí ở từng bước
- Nhược điểm:
  - Phụ thuộc vào các yêu cầu được xác định sớm từ đầu
  - Không khả thi trong một số trường hợp đòi hỏi có nhiều thay đổi
  - Trong thực tế, mỗi giai đoạn trong quy trình đều những cải tiến, phản hồi mới về các giai
  - đoạn trước đó, thường đòi hỏi phải sửa đổi, điều chỉnh các giai đoạn trước đó.

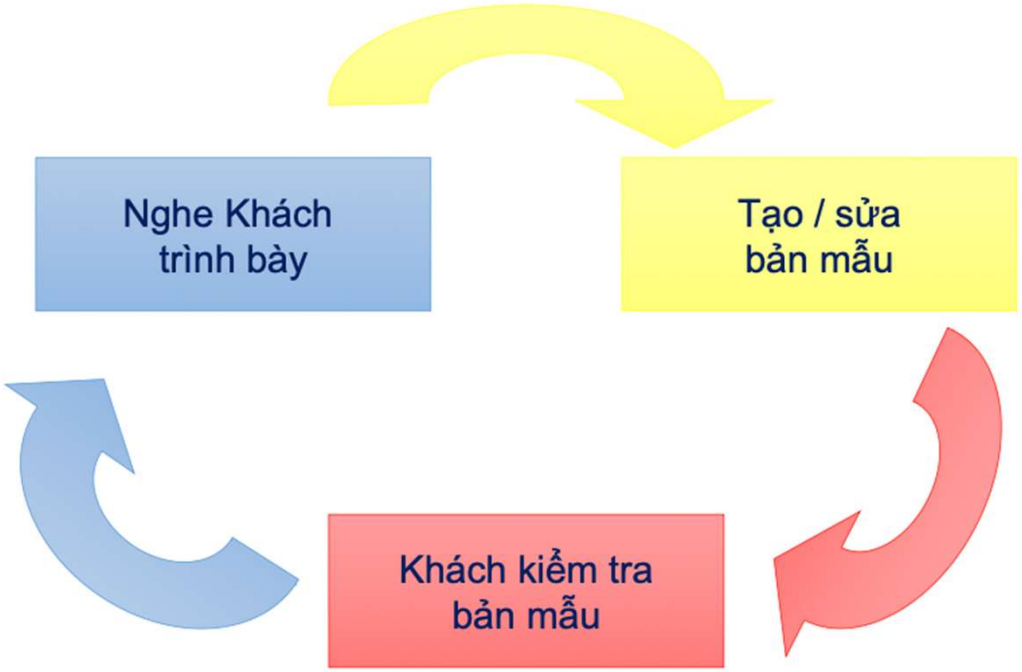
Quy trình phát triển PM

- **MH2. Mô hình thác nước sửa đổi/cải tiến**



Quy trình phát triển PM

- **MH3. Mô hình mẫu thử (prototype)**



Quy trình phát triển PM

- **MH3. Mô hình mẫu thử**
  - Khi mới rõ mục đích chung chung của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra
  - Dùng để thu thập yêu cầu qua các thiết kế nhanh
  - Các giải thuật, giải pháp kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là có mẫu để thảo luận gợi yêu cầu của người dùng

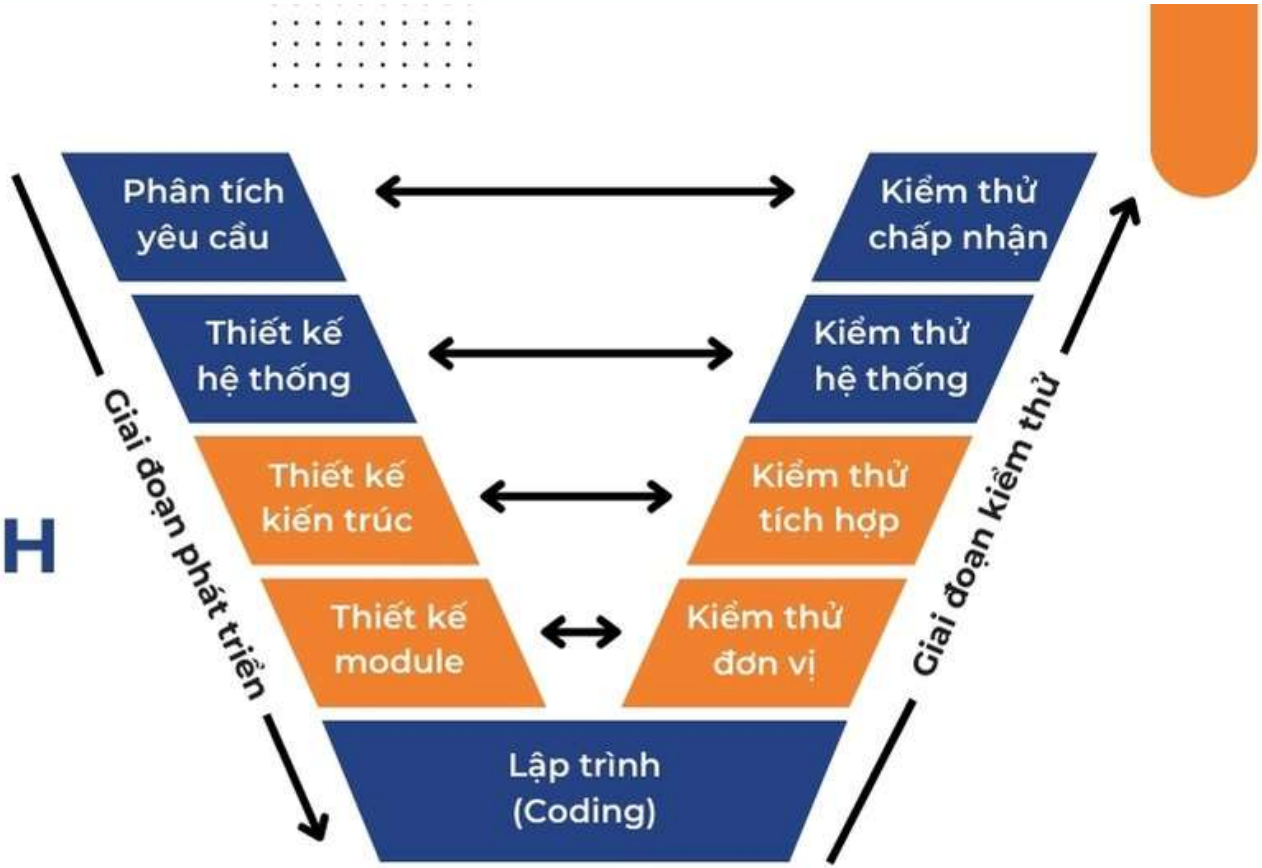
Quy trình phát triển PM

- **MH4. Mô hình V (Verification and Validation)**
  - Mỗi bước phát triển sẽ có bước **kiểm thử tương ứng**.
  - Nhấn mạnh kiểm thử ngay từ giai đoạn đầu.
- **Ưu điểm:**
  - Tăng độ tin cậy, chất lượng phần mềm.
  - Phát hiện lỗi sớm.
- **Nhược điểm:**
  - Cứng nhắc, ít linh hoạt khi thay đổi yêu cầu.

Quy trình phát triển PM

redefine management

MÔ HÌNH CHỮ V



02

Quy trình phát triển PM

- **MH5. Nhóm Mô hình tăng trưởng**
  - Phần lớn các hệ phần mềm phức tạp đều tiến hóa theo thời gian: môi trường thay đổi, yêu cầu phát sinh thêm, hoàn thiện thêm chức năng, tính năng
  - Các mô hình tiến hóa có tính lặp lại. Kỹ sư phần mềm tạo ra các phiên bản (versions) ngày càng hoàn thiện hơn, phức tạp hơn
  - Các mô hình tiêu biểu:
    - Gia tăng (Incremental)
    - Xoắn ốc (Spiral)
    - Xoắn ốc WINWIN (WINWIN spiral)
    - Phát triển đồng thời (Concurrent development)

Quy trình phát triển PM

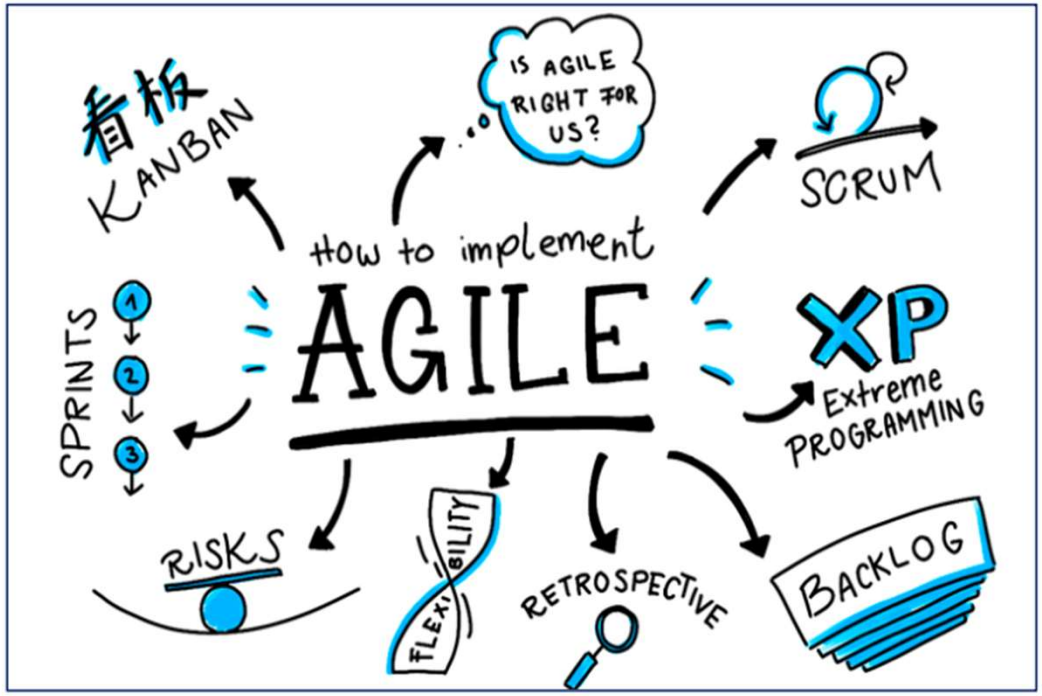
- **Triết lý Mô hình linh hoạt – Agile**
  - Phát triển phần mềm linh hoạt hoặc Lập trình linh hoạt (tiếng Anh: ***Agile software development*** hay ***Agile programming***)]
  - Agile Software Development là một thuật ngữ chung chỉ tất cả các kỹ thuật và phương pháp phát triển phần mềm theo triết lý Agile.
- Agile không phải là một công cụ hay một phương pháp duy nhất
  - Agile là triết lý được đưa ra vào năm 2001.
  - Từ triết lý, nguyên lý của Agile có thể áp dụng nhiều phương pháp khác nhau trong quy trình phát triển phần mềm
- Agile thay đổi đáng kể cách tiếp cận phát triển phần mềm
  - Hạn chế việc nặng theo hướng tài liệu (như mô hình thác nước).
  - Tương tác thông qua các quy trình và công cụ

Quy trình phát triển PM

- **Ý tưởng Agile:** ba ý tưởng chính được thảo luận:
  - Rút ngắn thời gian đưa sản phẩm ra thị trường (Speed to market)
  - Phản hồi nhanh chóng (Rapid feedback)
  - Cải tiến, cải thiện sản phẩm phần mềm liên tục (Continuous improvement)
- **Tuyên ngôn Agile:**
  - **Các cá nhân và tương tác** qua **các quy trình và công cụ**
  - **Phần mềm chạy tốt** hơn là **tài liệu đầy đủ/toàn diện**
  - **Cộng tác với khách hàng** hơn là **thương thảo hợp đồng**
  - **Đáp ứng với thay đổi** hơn là **bám sát kế hoạch**

Quy trình phát triển PM

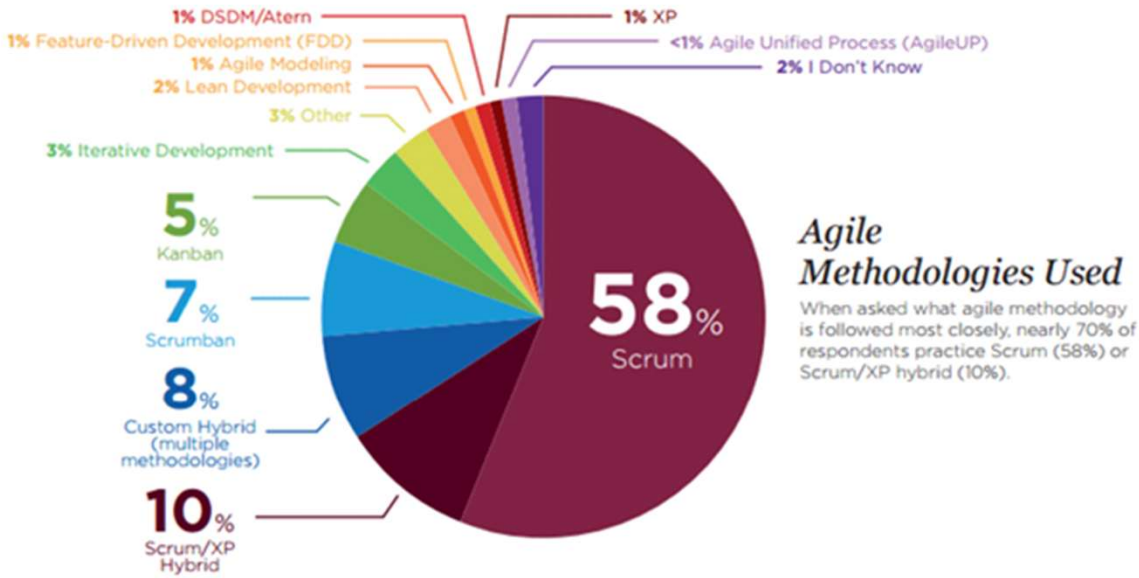
- **Cách tiếp cận Agile**
  - Agile không định nghĩa ra một phương pháp/cách thức cụ thể
  - Có nhiều phương pháp khác nhau thỏa mãn và hướng theo các tiêu chí của Agile
  - Thực tế: Nhiều công ty đã kết hợp các phương pháp lại với nhau



Hình 1.1: Agile – phương pháp phát triển phần mềm linh hoạt

Một số phương pháp theo mô hình Agile

- Scum
- Extreme programming (XP)
- Kanban
- Crystal methodology
- Feature-driven (FDD)
- ….

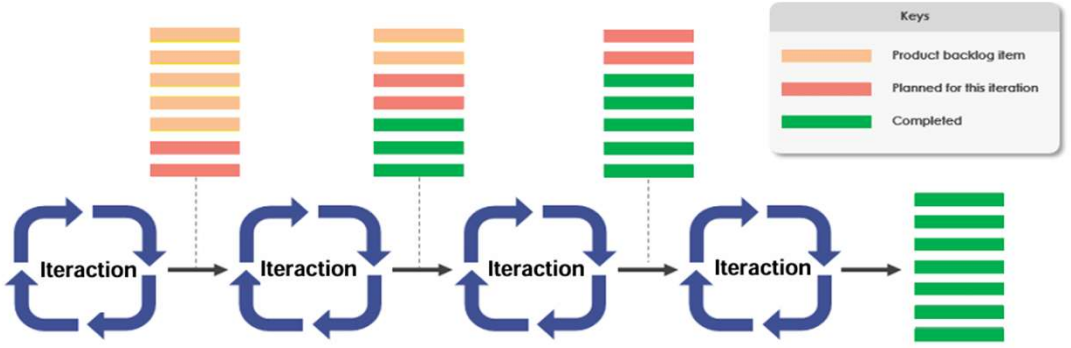


Biểu đồ: Mức độ ưa chuộng sử dụng các phương pháp Agile trong doanh nghiệp

Nguồn: Budiman, Thomas & Suroso, Jarot. (2017). Optimizing IT Infrastructure by Virtualization Approach. IOP Conference Series: Materials Science and Engineering.

Đặc trưng của Agile

- **Tính lặp (Iteration)**

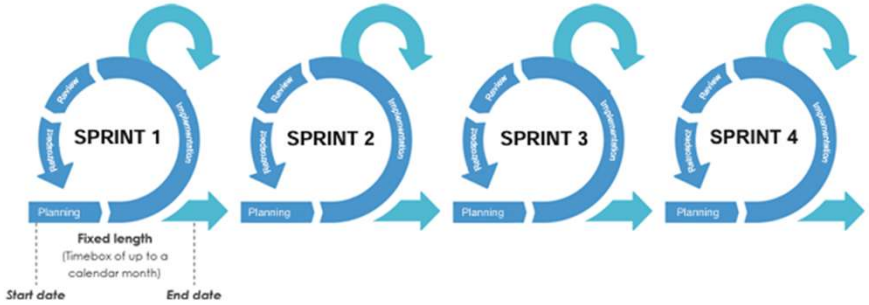


Sơ đồ: Các vòng lặp trong phương pháp Agile

- Dự án sẽ được thực hiện trong các phân đoạn lặp đi lặp lại (Iteration hoặc Sprint)
- Trong mỗi phân đoạn, thực hiện đầy đủ các công việc cần thiết như lập kế hoạch, phân tích yêu cầu, thiết kế, triển khai, kiểm thử.
- Khái niệm: Danh sách chức năng/công việc (Product baclog item) - danh sách các chức năng có thể được sắp xếp theo thứ tự ưu tiên mà một sản phẩm nên có.

Đặc trưng của Agile

- **Tính tăng trưởng (Increments & Evolutionary)**
- Mỗi bước lặp giống như phát triển một phần mềm hoàn chỉnh: xác định yêu cầu, phân tích thiết kế, viết mã, kiểm thử, viết tài liệu
- Kết quả sau mỗi giai đoạn phát triển đều được kiểm tra kĩ và sẵn sàng đưa vào sản phẩm
- Dự án được chia thành nhiều phần/ giai đoạn phát triển nhỏ (sprint) nối tiếp nhau, sau mỗi giai đoạn, hệ thống được từng bước hoàn thiện



Mô hình: Các bước lặp (sprint) của Scrum

Đặc trưng của Agile

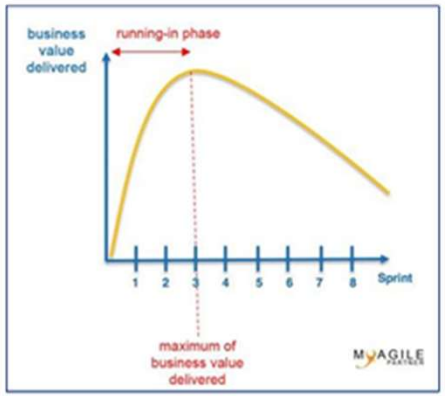
- **Tính thích ứng/thích nghi (Adaptive)**
- Kế hoạch sẽ liên tục được điều chỉnh
- Phù hợp theo các phân đoạn ngắn của dự án
- Kịp thời những yêu cầu thay đổi của khách hàng hay những tác động của các vấn đề khác
- **Nhóm tự tổ chức và liên chức năng**
- Nhóm tự tổ chức sẽ chịu trách nhiệm từng mảng công việc riêng biệt theo mỗi phân đoạn của dự án.
- Phù hợp với công việc được giao để có thể hoàn thành nhiệm vụ

Đặc trưng của Agile

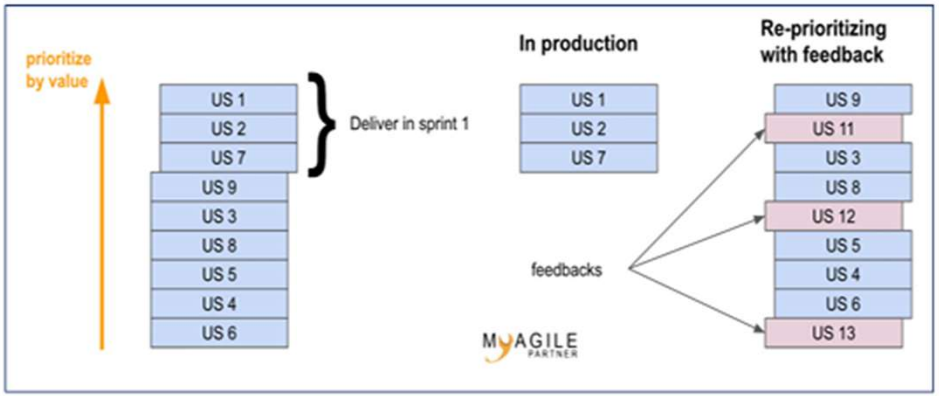
- **Quản lý tiến trình thực nghiệm (Empirical Process Control)**
- Dựa vào dữ liệu thực tế để đưa ra các quyết định cho công việc
- Rút ngắn thời gian phản hồi và tăng tính linh hoạt
- **Giao tiếp trực tiếp (Face-to-face communication)**
- Agile đánh giá cao việc trao đổi trực tiếp hơn là giao tiếp thông qua giấy tờ.
- Agile còn khuyến khích nhóm dự án trực tiếp nói chuyện với khách hàng để hiểu rõ điều họ đang cần.

Đặc trưng của Agile

- **Phát triển dựa trên giá trị (Value-based development)**
- Thường xuyên trao đổi với khách hàng
- Nắm được những yêu cầu có mức độ ưu tiên cao
- Đưa ra những điều chỉnh phù hợp nhằm đem lại giá trị sớm nhất



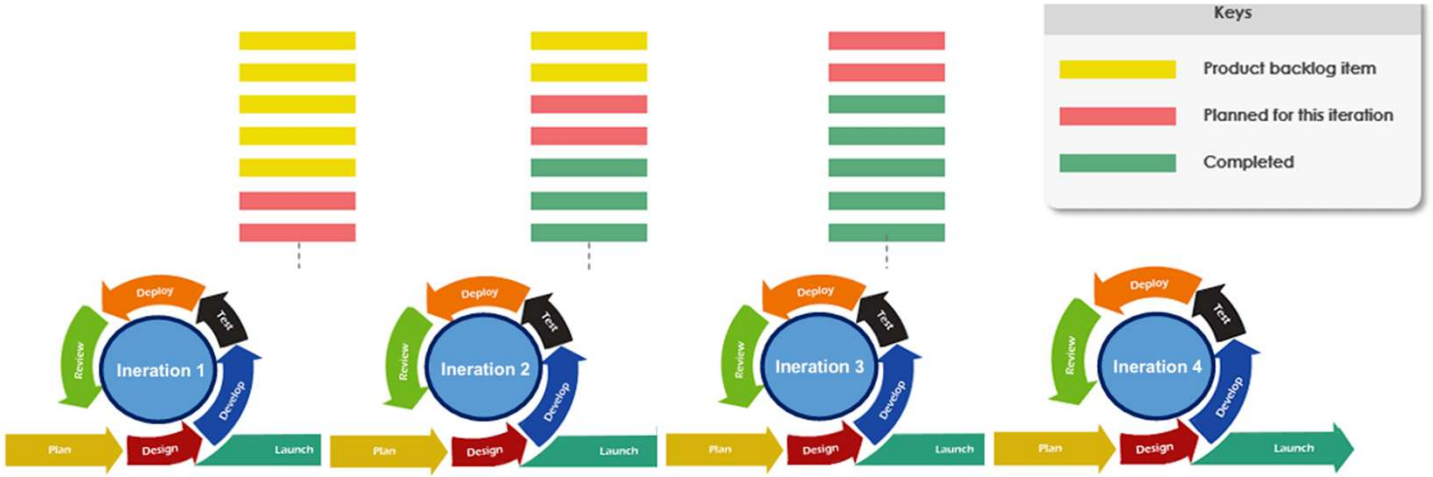
Biểu đồ: Value Driven-Development (VDD)



Biểu đồ: Value Driven-Development (VDD)

Phân tích quy trình Agile

- Trong mỗi phân đoạn, thực hiện đầy đủ các công việc cần thiết:
  - Lập kế hoạch, Phân tích thiết kế, Phát triển, Kiểm thử, Triển khai/bàn giao



Ví dụ mô hình phát triển theo phương pháp Agile

Ưu điểm của Agile

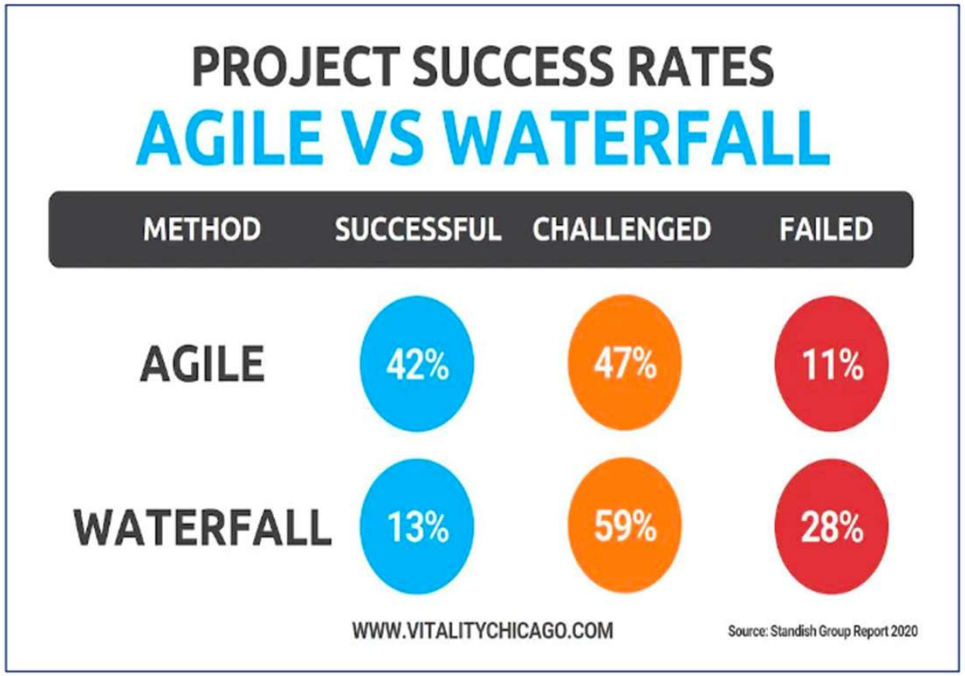
- Phù hợp với **những dự án nhỏ** thường có những **yêu cầu không được xác định rõ ràng** (có thể thay đổi thường xuyên).
- **Khách hàng** có thể được **xem trước từng phần dự án** trong quá trình phát triển, luôn sẵn sàng cho bất kỳ thay đổi từ phía khách hàng.
- Agile **chia dự án** thành những phần nhỏ và giao cho nhóm phát triển, hàng ngày tất cả mọi người phải họp trong khoảng thời gian ngắn để thảo luận về tiến độ và giải quyết những vấn đề nảy sinh nếu có nhằm đảm bảo đúng quy trình phát triển dự án.
- Tỉ lệ thành công của các dự án sử dụng Agile thường cao hơn các quy trình khác.

Ưu điểm của Agile

CHAOS RESOLUTION BY AGILE VS WATERFALL				
SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of software projects from FY2011-2015 within the new CHAOS database, segmented by the Agile process and waterfall method. The total number of software projects is over 10,000

Số liệu 2015



Số liệu 2020

Nhược điểm của Agile

- **Khó xác định** về loại dự án phần mềm nào phù hợp nhất cho cách tiếp cận Agile
- Nhiều tổ chức lớn gặp khó khăn trong việc chuyển từ phương pháp truyền thống sang một phương pháp Agile (linh hoạt).
- Khi Agile có rủi ro:
  - Phát triển **quy mô lớn** (> 20 nhà phát triển)
  - Phát triển **phân tán** (các nhóm không nằm chung)
  - Khách hàng hoặc **người liên hệ không đáng tin cậy**
  - Bắt buộc **quy trình nhanh** trong nhóm phát triển
  - Nhà phát triển **thiếu kinh nghiệm**

Nguyên lý SOLID

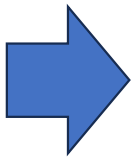
- SOLID “Principles Of OOD”, Robert C. Martin (“Uncle BOB”)
- Mục tiêu:
  - Viết code dễ hiểu, dễ bảo trì, dễ mở rộng.
  - Giảm sự phụ thuộc và lỗi khi thay đổi.
- SOLID bao gồm:
  - S – Single-responsiblity principle
  - O – Open-closed principle
  - L – Liskov substitution principle
  - I – Interface segregation principle
  - D – Dependency Inversion Principle

Open-closed Principle

- Các thực thể phần mềm (lớp, module, phương thức, …) nên là MỞ cho các mở rộng nhưng ĐÓNG cho các sửa đổi (open for extension, but closed for modification)
- “Open for extension”: một module (class) phải cung cấp các điểm mở rộng, cho phép thay đổi hành vi của nó
- Closed for modification”: Mã nguồn của module không cần thay đổi để cài đặt sự mở rộng đó

Open-closed Principle

```
class AreaCalculator {
    double calculate(Object shape) {
        if (shape instanceof Circle) {
            Circle c = (Circle) shape;
            return Math.PI * c.radius *
c.radius;
        }
        if (shape instanceof Rectangle) {
            Rectangle r = (Rectangle)
shape;
            return r.width * r.height;
        }
        return 0;
    }
}
```



```
interface Shape {
    double area();
}
class Circle implements Shape {
    double radius;
    public double area() { return Math.PI * radius *
radius; }
}
class Rectangle implements Shape {
    double width, height;
    public double area() { return width * height; }
}
class AreaCalculator {
    double calculate(Shape shape) {
        return shape.area();
    }
}
```

Liskov Substitution Principle

- Subclass phải có thể thay thế superclass mà **không phá vỡ logic**

```
class Bird {
    void fly() {
System.out.println("Flying"); }
}

class Ostrich extends Bird {
    @Override
    void fly() { throw new
UnsupportedOperationException("O
strich can't fly"); }
}
```

```
interface Bird { }

interface FlyingBird extends Bird {
    void fly();
}

class Sparrow implements FlyingBird
{
    public void fly() {
System.out.println("Flying"); }
}

class Ostrich implements Bird {
    // Không implement fly
}
```

Interface Segregation Principle

- Không ép class implement interface có method **không dùng**

```
interface Worker {
    void work();
    void eat();
}

class Robot implements Worker {
    public void work() { /* làm việc */ }
    public void eat() { /* ??? Robot
không ăn */ }
}
```

```
interface Workable {
    void work();
}
interface Eatable {
    void eat();
}
class Human implements Workable,
Eatable {
    public void work() { /* làm việc */ }
    public void eat() { /* ăn */ }
}
class Robot implements Workable {
    public void work() { /* làm việc */ }
}
```

Dependency Inversion Principle

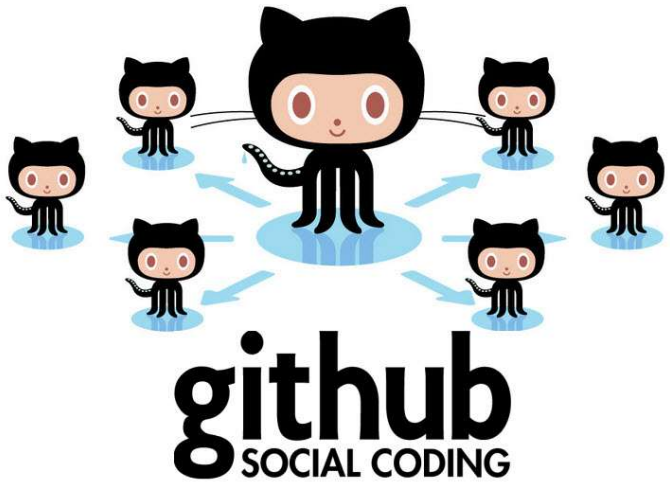
- Module cấp cao không nên phụ thuộc vào module cấp thấp.
- Cả hai nên phụ thuộc vào abstraction (interface).

```
class MySQLDatabase {
    void connect() { /* kết nối MySQL */ }
}

class UserService {
    MySQLDatabase db = new MySQLDatabase();
}
```

```
interface Database {
    void connect();
}
class MySQLDatabase implements Database {
    public void connect() { /* kết nối MySQL */ }
}
class UserService {
    private Database db;
    public UserService(Database db) {
this.db = db; }
}
```

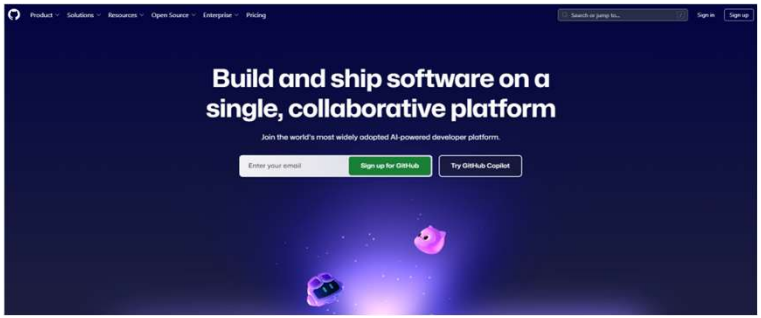
Quản lý phiên bản phần mềm

- Git: Hệ thống quản lý phiên bản phân tán (Distributed Version Control System)
- GitHub: Nền tảng lưu trữ mã nguồn trực tuyến dựa trên Git, hỗ trợ cộng tác
- Lợi ích:
  - Theo dõi lịch sử thay đổi.
  - Làm việc nhóm hiệu quả
  - Tích hợp CI/CD, quản lý dự án



GitHub

- **Cài đặt & Tạo tài khoản**
- Cài Git:
  - git-scm.com → Tải & cài đặt.
  - Kiểm tra: git --version.
- Tạo tài khoản GitHub:
  - Truy cập github.com.
  - Đăng ký và xác thực email.




```
git config --global user.name “your name"
git config --global user.email
"email@example.com"
```

Một số khái niệm cơ bản

- Repository (Repo): Nơi lưu trữ mã nguồn.
- Commit: Lưu một phiên bản mới của mã.
- Branch: Nhánh phát triển song song.
- Merge: Gộp thay đổi từ nhánh này sang nhánh khác.
- Clone: Tải repo từ GitHub về máy.
- Push/Pull: Đồng bộ giữa local và GitHub.

Thao tác cơ bản

- Clone hoặc Init repo
- Chỉnh sửa mã
- git add → thêm file vào staging
- git commit → lưu phiên bản
- git push → đưa lên GitHub
- Khi cần cập nhật: git pull

```
git add main.java
git commit -m "Thêm chức năng đăng nhập"
git push origin main
```

Làm việc nhóm trên GitHub

- **Fork**: Sao chép repo của người khác về tài khoản mình.
- **Pull Request (PR)**:
  - Gửi yêu cầu hợp nhất code vào repo gốc.
  - Thường dùng khi đóng góp cho dự án open-source.
- **Code Review**: Thành viên nhóm xem xét code trước khi merge.
- *Lưu ý*: Sử dụng branch riêng cho từng tính năng.

Kỹ nghệ

- Commit nhỏ, thông điệp rõ ràng
- Thường xuyên git pull để tránh xung đột
- Dùng .gitignore để bỏ qua file không cần thiết
- Tạo README.md để giới thiệu dự án
- Backup code bằng cách push thường xuyên
- Câu hỏi thảo luận: Khi nào nên tạo branch mới?
  Xử lý xung đột merge thế nào?

THANK YOU !