Trường Đại học Công Nghệ Thông Tin, ĐHQG-HCM
Khoa Công Nghệ Phần Mềm

Chương 4: Vòng Đời Tồn Tại Phát Triển Hệ Thống

Giảng viên: ThS. Lê Thanh Trọng

NỘI DUNG

1. Hệ thống thông tin
2. Vòng đời tồn tại phát triển hệ thống (SDLC)
3. Các mô hình phát triển phần mềm
   - Mô hình thác nước
   - Mô hình lặp
   - Mô hình "V"
   - Mô hình xoắn ốc
   - Mô hình Agile
   - Mô hình RAD
   - Mô hình Prototype

NỘI DUNG

1. **Hệ thống thông tin**
2. Vòng đời tồn tại phát triển hệ thống (SDLC)
3. Các mô hình phát triển phần mềm
   - Mô hình thác nước
   - Mô hình lặp
   - Mô hình "V"
   - Mô hình xoắn ốc
   - Mô hình Agile
   - Mô hình RAD
   - Mô hình Prototype

THÔNG TIN

- Thông tin là **tài nguyên** của tổ chức mà cần được quản lý cẩn thận như những nguồn tài nguyên khác
- Giá thành liên kết với quá trình xử lý thông tin
- Xử lý thông tin cần phải được quản lý nắm bắt tất cả ưu điểm tiềm năng của nó

CÁC DẠNG CỦA HỆ THỐNG

- Hệ thống **xử lý giao dịch** (Transaction Processing Systems - TPS)
  - Sử dụng cho số lượng lớn dữ liệu cho các giao dịch hằng ngày ví dụ như giao dịch tài chính của công ty (giảm thời gian thực hiện các tác vụ)
- Hệ thống **tự động văn phòng** (Office Automation Systems - OAS)
  - Hỗ trợ người làm việc mà không thường xuyên tạo kiến thức mới như xử lý văn bản, email…
- Hệ thống làm việc **tri thức** (Knowledge Work Systems - KWS)
  - Giúp đỡ chuyên nghiệp trong sự cố gắng tạo kiến thức mới như nhà khoa học, kỹ sư

CÁC DẠNG CỦA HỆ THỐNG

- Hệ thống **thông tin quản lý** (Management Information Systems MIS)
  - Bao gồm hệ thống xử lý giao dịch
  - Đầu ra sử dụng trong sự đưa ra quyết định
- Hệ thống **hỗ trợ quyết định**
  - Giống như MIS và cả hai phụ thuộc vào CSDL như nguồn dữ liệu
  - Khác biệt hơn MIS bởi vì nó làm nổi bật sự hỗ trợ đưa ra quyết định
  - Quyết định thật sự vẫn được làm bởi người sử dụng
  - Gần gũi nhu cầu hơn tới người hoặc nhóm sử dụng hệ thống hơn các hệ thống cổ điển

CÁC DẠNG CỦA HỆ THỐNG

- Hệ **chuyên gia và trí tuệ nhân tạo** (Expert systems (ES) & Artificial Intelligence)
  - Hiệu quả chiến lược và sử dụng kiến thức chuyên gia
  - Thành phần bao gồm cơ sở tri thức, máy suy diễn và giao diện người sử dụng
- Hệ thống **hỗ trợ thực hiện** (Executive Support Systems - EES)
  - Tương tác với môi trường bên ngoài
  - Cung cấp đồ hoạ
  - Trả lời thông tin tập hợp bởi TPS & MIS

ẢNH HƯỞNG CÔNG NGHỆ MỚI

- Công nghệ mới kết hợp vào bên trong hệ thống cổ điển
  - Thương mại điện tử sử dụng Web thực hiện hoạt động nghiệp vụ
  - ERP (Enterprise Resource Planning) có mục đích kết hợp nhiều hệ thống thông tin khác nhau bên trong tập đoàn
  - Wireless và thiết bị cầm tay bao gồm sản phẩm di động
  - Phần mềm mã nguồn mở

SỰ CẦN THIẾT PHÂN TÍCH VÀ THIẾT KẾ

- Phân tích và thiết kế hệ thống là một phương pháp hệ thống
  - Chỉ ra vấn đề, cơ hội và mục tiêu
  - Phân tích dòng thông tin trong tổ chức
  - Thiết kế hệ thống thông tin máy tính giải quyết vấn đề
- Sự tham gia của người sử dụng hệ thống là rất quan trọng

PHÂN TÍCH HỆ THỐNG

- Phân tích hệ thống hành động như sau:
  - Tư vấn bên ngoài tới nghiệp vụ
    - Thuê địa sản phẩm bên ngoài
    - Cung cấp triển vọng mới
  - Hỗ trợ chuyên gia bên trong hệ thống
    - Phục vụ như nguồn tài nguyên bên trong tổ chức
  - Như một tác nhân thay đổi
    - Dễ dàng thay đổi sử dụng HTTT
    - Phải tương tác với người sử dụng và quản lý
    - Phát triển kế hoạch đối với sự thay đổi và luôn luôn tương tác với tác nhân tạo ra thay đổi

PHÂN TÍCH HỆ THỐNG

- Phân tích là người giải quyết vấn đề về yêu cầu, và yêu cầu kỹ năng giao tiếp
- Phân tích hệ thống sẽ chỉ ra vấn đề cần xem xét như thử thách và giải pháp giải quyết
- Phân tích cần phải đúng nguyên tắc xử lý với người sử dụng và khách hàng
- Phân tích hệ thống cần tự bản thân rèn luyện và tự thức đẩy bản thân, thực hiện tốt việc quản lý và sắp xếp tài nguyên dự án bao gồm cả con người

NỘI DUNG

1. Hệ thống thông tin
2. **Vòng đời tồn tại phát triển hệ thống (SDLC)**
3. Các mô hình phát triển phần mềm
   - Mô hình thác nước
   - Mô hình lặp
   - Mô hình "V"
   - Mô hình xoắn ốc
   - Mô hình Agile
   - Mô hình RAD
   - Mô hình Prototype

SDLC LÀ GÌ?

- System Development Life Cycle: Là một quy trình được sử dụng bởi ngành công nghiệp phần mềm để thiết kế
- Mục đích: sản xuất một phần mềm **chất lượng** cao, đáp ứng hoặc vượt quá **mong đợi của khách hàng**, hoàn thành trong **thời gian** và ước tính **chi phí**
- Là một khung xác định các nhiệm vụ được thực hiện ở mỗi bước trong quy trình phát triển

SDLC

- Bao gồm một kế hoạch chi tiết mô tả cách phát triển, bảo trì, thay thế và thay đổi hoặc nâng cao phần mềm cụ thể
- Vòng đời xác định một phương pháp để cải thiện chất lượng phần mềm và quy trình phát triển tổng thể
- ISO/IEC 12207 là một tiêu chuẩn quốc tế nhằm mục đích trở thành tiêu chuẩn xác định tất cả các nhiệm vụ cần thiết để phát triển và bảo trì phần mềm

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

- SDLC là phương pháp hệ thống giải quyết vấn đề nghiệp vụ
- Nó được chia làm **7 bước**
- Mỗi bước có những hoạt động duy nhất
- Ở đó có thường có sự chồng chéo của các bước
- Các hoạt động **có thể lặp lại**
- Những hoạt động khác nhau có thể xuất hiện tại cùng thời điểm

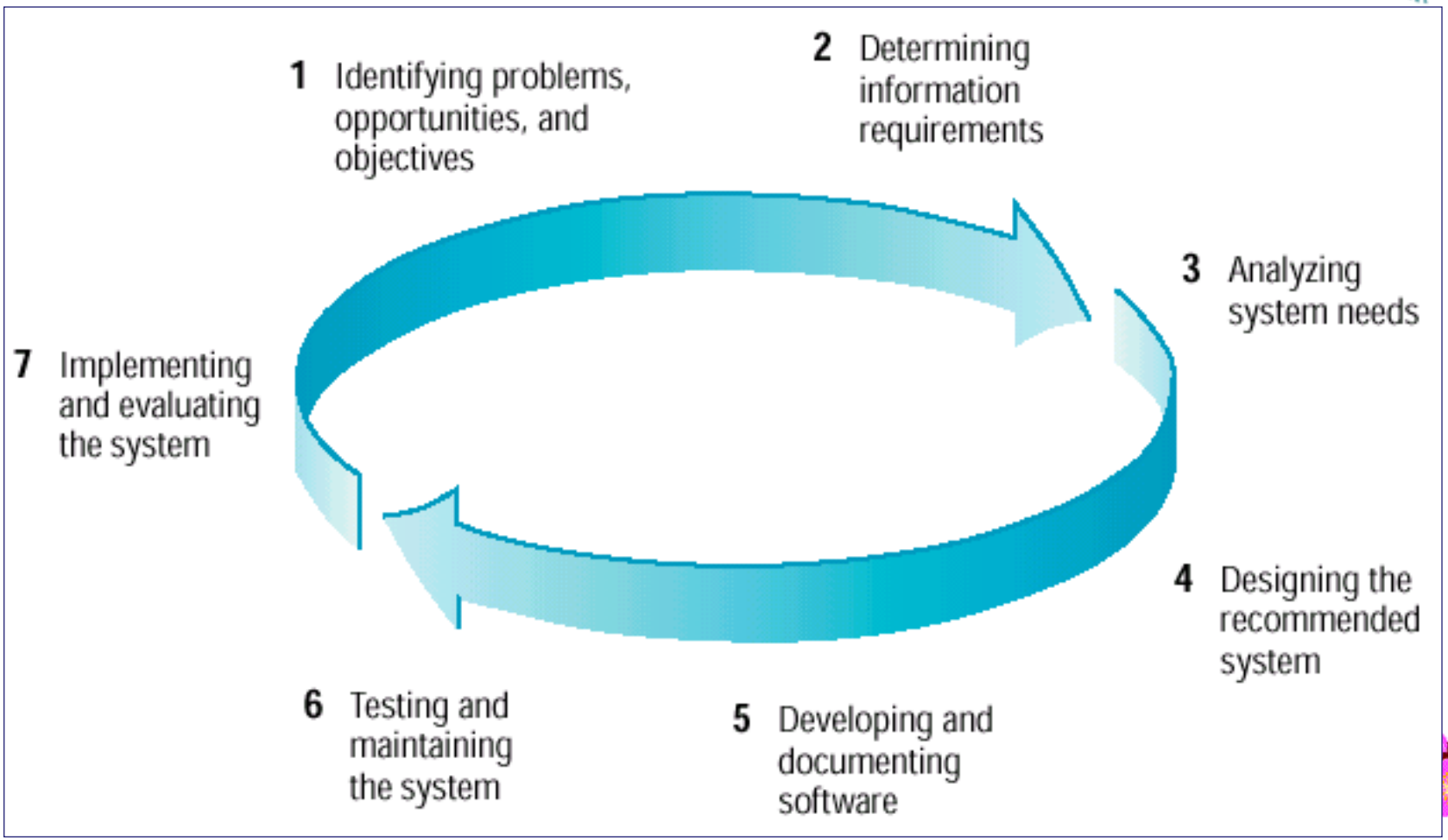
CÁCH TIẾP CẬN

- Có hai cách tiếp cận để củng cố các vòng đời phát triển của hệ thống:
  - **Cách tiếp cận tuyến tính:** các bước được thực hiện theo trình tự và được hoàn thành trước khi chuyển sang
  - **Cách tiếp cận tiến hóa:** giải pháp được phân phối xuất hiện trong các giai đoạn phát truể phần mềm lặp lại

MỘT SỐ VẤN ĐỀ

- **Yêu cầu có thể không đầy đủ** trong bước lấy yêu cầu (người dùng mới, nghiệp vụ mới có thể chưa định rõ yêu cầu,..)
- Sự **phát triển của công nghệ, nghiệp vụ** rất nhanh, cần có sự linh hoạt, nhạy bén để giúp việc xây dựng hiệu quả, tiết kiệm

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)



VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

❖ Bước 1: Lập kế hoạch

- Xác định phạm vi, yêu cầu liên quan đến việc nghiên cứu hệ thống hiện có và thu thập thông tin chi tiết để tìm hiểu các yêu cầu là gì, cách thức hoạt động và nơi cần cải thiện,…
- Thực hiện phỏng vấn quản lý người sử dụng nhằm xác định tính khả thi sản phẩm
- **Kết thúc bước:** báo cáo tính khả thi chứa đựng định nghĩa một vấn đề và tổng quát hoá mục tiêu, phạm vi và các rủi ro liên quan

Phân tích tính khả thi

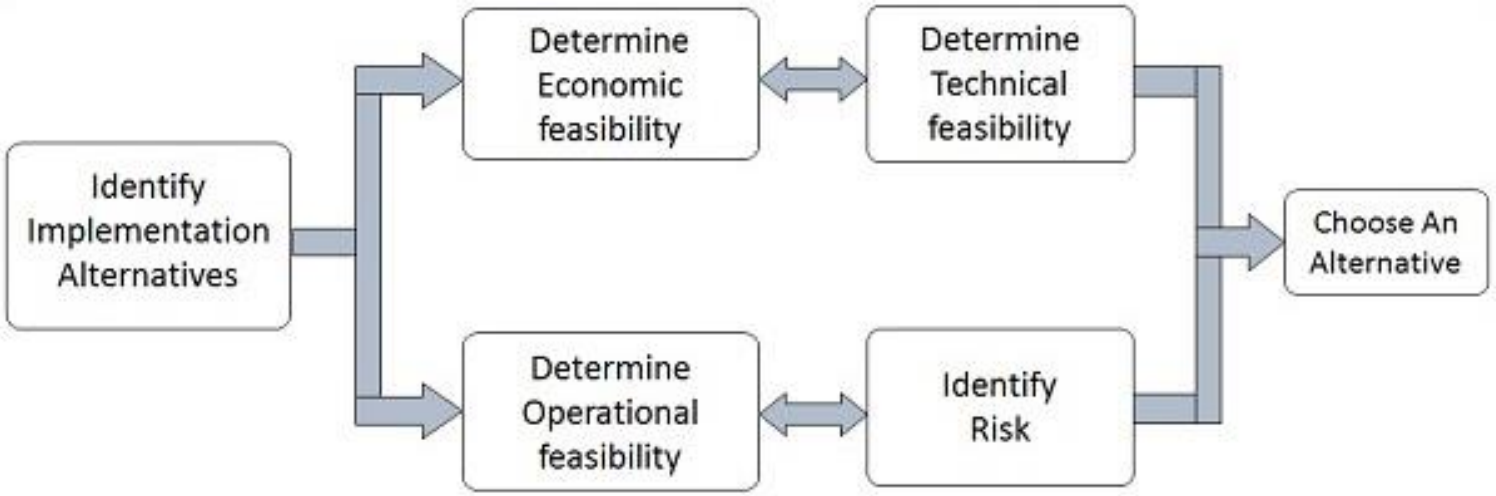
- Điều tra sơ bộ giúp ban quản lý đưa ra quyết định về việc liệu nghiên cứu hệ thống có khả thi để phát triển hay không
- Xác định khả năng cải thiện một hệ thống hiện có, phát triển một hệ thống mới và đưa ra các ước tính cụ thể cho phát triển hệ thống
- Tạo các phác thảo của vấn đề và quyết định xem giải pháp khả thi hay phù hợp có tồn tại hay không
- Mục tiêu chính là thu được **phạm vi vấn đề** và **đề xuất hệ thống chính**

Các bước phân tích khả thi

- Thành lập nhóm dự án và một người lãnh đạo dự án
- Xây dựng sơ đồ hệ thống
- Xác định những thiếu sót của hệ thống hiện tại và đặt ra mục tiêu
- Tìm giải pháp thay thế hoặc hệ thống ứng viên tiềm năng
- Xác định tính khả thi kỹ thuật, tính khả thi trong vận hành, khả thi về mặt kinh tế,…
- Cân nhắc hiệu suất và hiệu quả chi phí
- Xếp hạng các lựa chọn và chọn ứng cử viên tốt nhất
- Chuẩn bị một đề xuất hệ thống cuối cùng để nhà quản lý phê duyệt

Các khía cạnh tính khả thi

- **Khả thi về kinh tế:** chi phí, lợi nhuận ròng, tính đầu tư, hoàn vốn, tối đa giá trị, rủi ro thất thoát,…
- **Khả thi kĩ thuật:** công nghệ hiện có, nâng cấp kĩ thuật,…
- **Khả thi vận hành:** hiệu quả khi triển khai, phù hợp với tổ chức, tính tương thích, tài nguyên máy tính, mạng,…
- **Khả thi về hành vi:** thái độ và hành vi người dung, công tác đào tạo, chuyển giao, thay đổi công việc/nghiệp vụ,…



VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

❖ Bước 2: Xác định yêu cầu

- Người sử dụng cần gì để thực hiện công việc đó
- Phỏng vấn quản lý, nhân viên thực hiện
- Tập hợp hệ thống/quản lý tài liệu
- Sử dụng bản câu hỏi, form
- Quan sát hệ thống và cá nhân tham gia
- Joint Application Development (JAD): workshop cho owners, users, analysts, designers và builders
- **Kết thúc bước:** tài liệu SRS (Đặc tả yêu cầu phần mềm-Software Requirements Specification) bao gồm tất cả các yêu cầu sản phẩm được thiết kế và phát triển trong vòng đời dự án

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

❖ Bước 3: Phân tích

- Tạo sơ đồ dòng dữ liệu
- Dẫn chứng tài liệu lập luận logic thủ tục cho quá trình xủ lý sơ đồ dòng dữ liệu
- Hoàn thành từ điển dữ liệu
- Chuẩn bị và trình bày mô hình hệ thống
- Đề nghị giải pháp tối ưu để quản lý
- **Kết thúc bước:** Mô hình hệ thống mà tổng quát tìm kiếm cái gì, cung cấp giá thành/lợi ích trong quá trình chọn lựa và đưa ra những đề nghị sẽ làm gì trong mô hình

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

Bước 4: Thiết kế

- Thiết kế giao diện sử dụng
  - Thiết kế đầu ra
  - Thiết kế đầu vào
- Thiết kế kiểm tra dữ liệu
- Thiết kế file, CSDL
- Tạo các đặc trưng chương trình
- Tạo các cây hoặc bảng quyết định
- **Kết thúc bước:** Thiết kế bên trong của tất cả các mô-đun của hệ thống (Design Document Specification-DDS)

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

❖ Bước 5: Xây dựng hệ thống

- Công nghệ và ngôn ngữ lập trình được chọn liên quan đến loại phần mềm
- Viết mã cho các thiết kế theo DDS
- Phải tuân theo các nguyên tắc mã hóa được xác định bởi các công cụ tổ chức và lập trình của họ như trình biên dịch, trình thông dịch, trình gỡ lỗi,...
- Các ngôn ngữ lập trình cấp cao khác nhau như C, C++, Pascal, Java, Python,…

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

❖ Bước 6: Thử nghiệm và bảo trì hệ thống

- Sự thử nghiệm được tiến hành bởi lập trình viên, phân tích hệ thống, đội kiểm tra chất lượng
- Dữ liệu đơn giản được tạo và khi đó sự thử nghiệm tiến hành trên dữ liệu
- Sự bảo trì hệ thống và tài liệu của nó bắt đầu trong bước này như vấn đề được tìm kiếm
- Thực hiện cho đến khi sản phẩm đạt tiêu chuẩn chất lượng được xác định trong SRS

VÒNG ĐỜI TỒN TẠI PHÁT TRIỂN HỆ THỐNG (SDLC)

Bước 7: Vận hành và đánh giá hệ thống

- Đặt kế hoạch chuyển đổi
- Huấn luyện người sử dụng
- Mua và cài đặt thiết bị mới
- Chuyển hoá các file từ hệ thống cũ sang hệ thống mới
- Cài đặt hệ thống
- Xem xét và đánh giá hệ thống

BẢO TRÌ HỆ THỐNG

- Bảo trì hệ thống là:
  - **Loại bỏ lỗi** không phát hiện được
  - **Nâng cấp** phần mềm tồn tại
- Thời gian bảo hành chiếm tỉ lệ khoảng từ 48%-60% tổng thời gian
- Hệ thống được nâng cấp trong những **nguyên nhân** sau:
  - Cộng thêm các đặc tính bổ sung vào hệ thống
  - Thay đổi nghiệp vụ theo thời gian
  - Thay đổi trong công nghệ, phần cứng và phần mềm

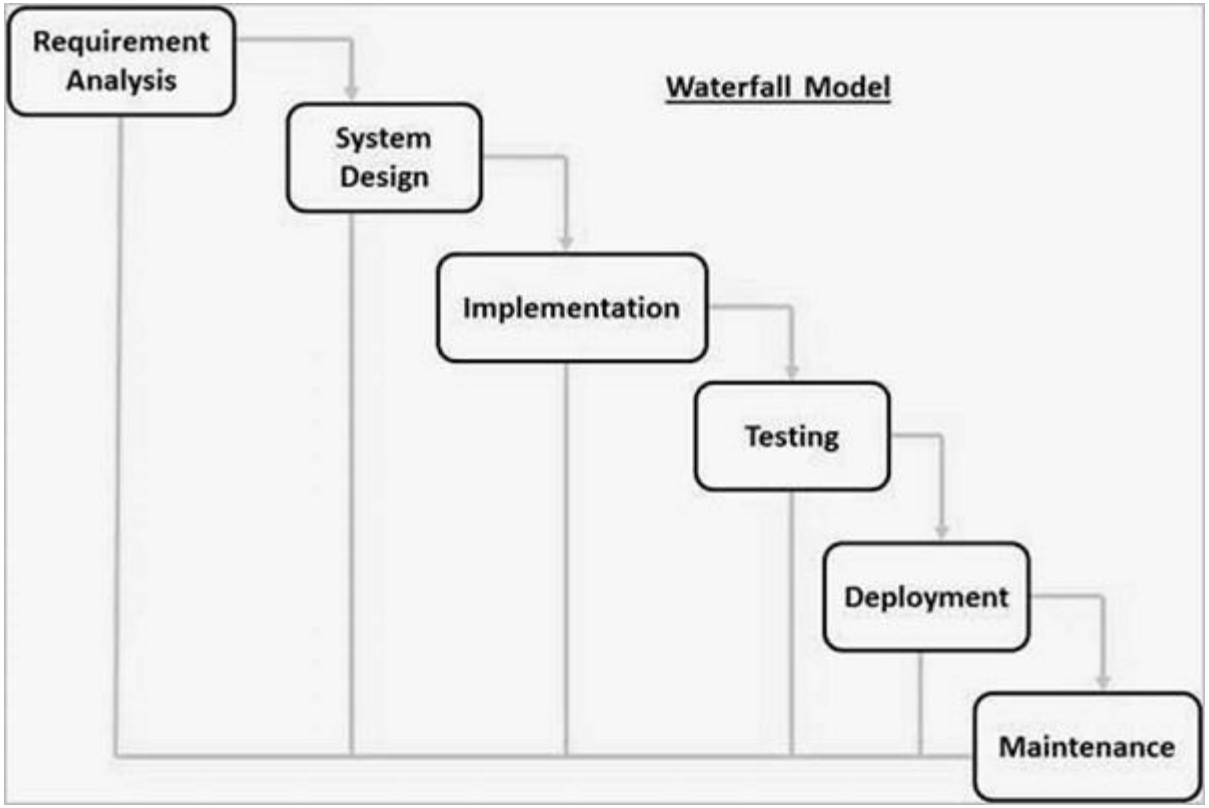
NỘI DUNG

1. Hệ thống thông tin
2. Vòng đời tồn tại phát triển hệ thống (SDLC)
3. **Các mô hình phát triển phần mềm**
   - Mô hình thác nước
   - Mô hình lặp
   - Mô hình "V"
   - Mô hình xoắn ốc
   - Mô hình Agile
   - Mô hình RAD
   - Mô hình Prototype

MÔ HÌNH THÁC NƯỚC

- Một cái giai đoạn chỉ được bắt đầu khi **giai đoạn trước nó đã kết thúc và xác nhận**
- Trong quá trình hoạt động nếu có vấn đề thì phải lập tức **review**
- Dự án có thể **kéo dài rất lâu** vì các giai đoạn có ảnh hưởng liên đới và phụ thuộc nhau
- Không thích hợp với sự thay đổi thường xuyên

MÔ HÌNH THÁC NƯỚC



TRƯỜNG HỢP SỬ DỤNG

- Yêu cầu rõ ràng, không thay đổi
- Công nghệ và sản phẩm cố định
- Các thông tin, tài nguyên được cung cấp sẵn
- Dự án nhỏ

ƯU ĐIỂM

- Đơn giản và dễ áp dụng
- Mỗi pha hoàn thành ở một thời điểm nhất định
- Phù hợp với dự án nhỏ và yêu cầu rõ rang
- Các pha biệt lập, rõ ràng
- Các cột mốc thời gian được phân định rõ, không trùng lắp
- Dễ sắp xếp kế hoạch và công việc
- Quy trình và kết quả được văn bản hóa dễ dàng

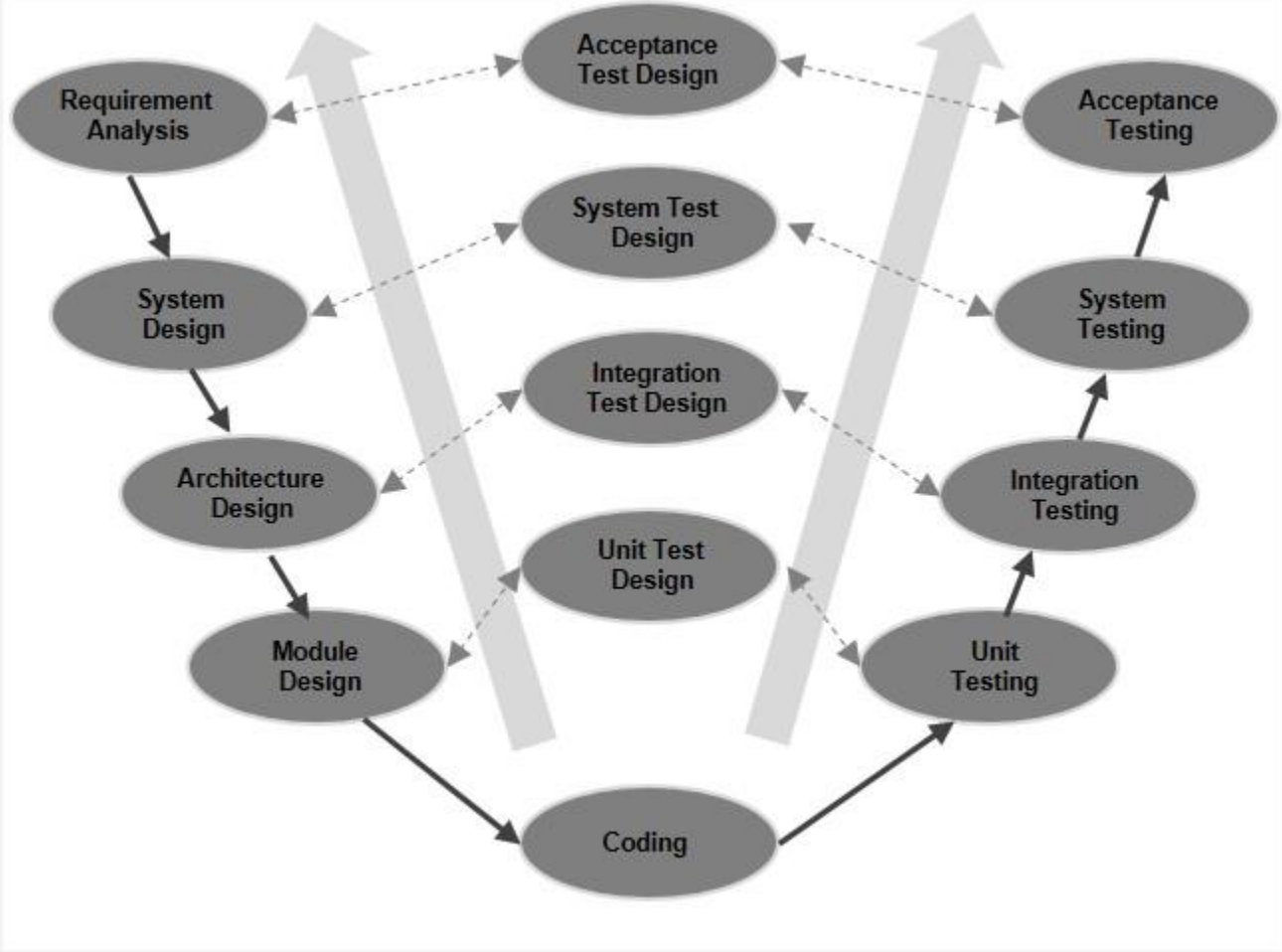
NHƯỢC ĐIỂM

- Chỉ tiếp cận sản phẩm ở giai đoạn cuối
- Tính rủi ro và bất định cao (ko rõ sản phẩm cuối cùng match với yêu cầu thế nào)
- Không phù hợp cho hệ thống phức tạp và phương pháp hướng đối tượng
- Khó điều tiết, hiệu chỉnh vơi sự thay đổi yêu cầu
- Hiệu chỉnh phạm vi có thể dẫn đến kết thúc dự án

MÔ HÌNH "V"

- Các hoạt động trong mô hình được tiến hành theo dạng chữ "V"
- Tương tự như mô hình thác nước nhưng mỗi giai đoạn đều kết hợp với hoạt động kiểm tra

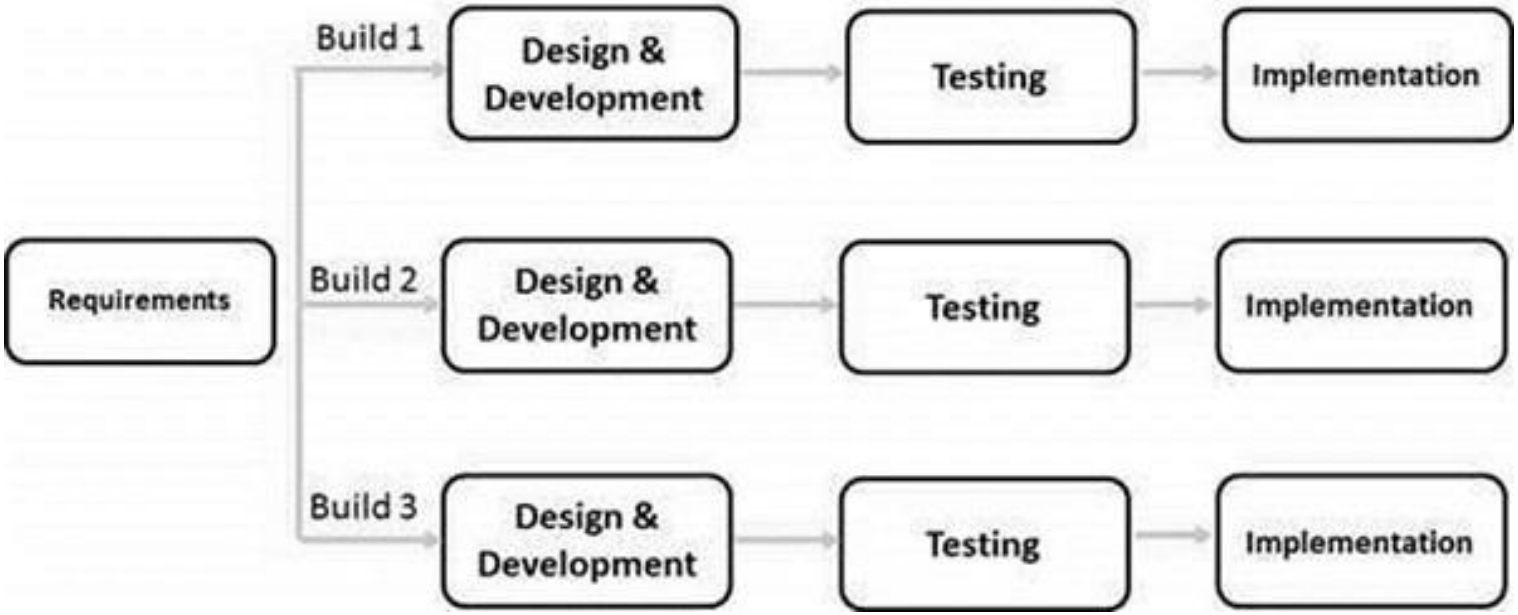
MÔ HÌNH "V"


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

MÔ HÌNH LẶP

- Xây dựng nhanh ứng dụng con **đáp ứng một phần yêu cầu**
- Ứng dụng con cũng được kiểm tra, đánh giá và hiệu chỉnh
- Trong mỗi bước lặp, **mở rộng ứng dụng con** lớn dần cho đến khi đáp ứng tất cả yêu cầu
- Có thể có nhiều vòng lặp tại một thời điểm và ứng dụng lớn có thể được **chia thành nhiều hiện thực con** (build)
- Hoạt động kiểm thử cần được tiến hành nghiêm túc, hiệu quả

MÔ HÌNH LẶP


Requirements
Build 1
Design & Development
Testing
Implementation
Build 2
Design & Development
Testing
Implementation
Build 3
Design & Development
Testing
Implementation

NGỮ CẢNH ÁP DỤNG

- Yêu cầu của toàn bộ được xác định rõ ràng
- Các yêu cầu chính phải được xác định đúng
- Dự án chịu ràng buộc về thời gian bàn giao
- Thử nghiệm công nghệ mới
- Tài nguyên và kĩ năng cần thiết cho mỗi bước lặp cần được xác định rõ và chuẩn bị trước
- Thích ứng với sự thay đổi

ƯU ĐIỂM

- Một số chức năng được hiện thực nhanh
- Thực hiện song song các chức năng
- Quá trình thực hiện có thể đo lường hiệu quả dễ dàng
- Chi phí thấp khi có sự thay đổi (yêu cầu, phạm vi)
- Dễ kiểm tra, gở lỗi
- Phân tích, phát hiện và quản lý các rủi ro dễ dàng
- Thời gian thực hiện hoạt động khởi đầu ngắn
- Phù hợp cho dự án lớn và cấp bách
- Tận dụng được phản hồi của khách hang trong suốt quá trình phát triển

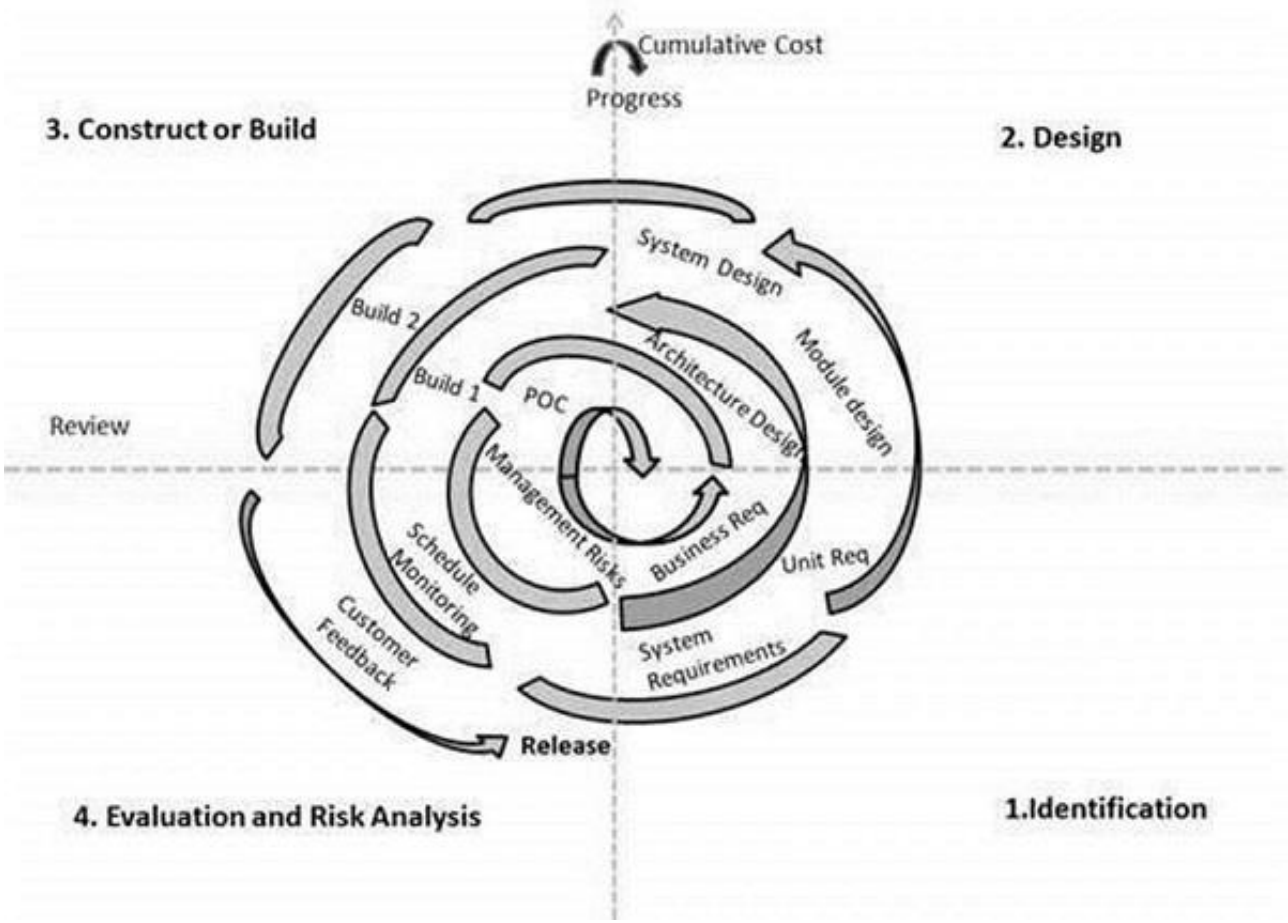
NHƯỢC ĐIỂM

- Cần nhiều tài nguyên
- Công tác quản lý cần được chú trọng hơn
- Phát thảo kiến trúc và thiết kế kiến trúc liên tục
- Định sự cải tiến rõ ràng, liên tục trong mỗi bước lặp
- Không thích hợp cho dự án nhỏ
- Cần nhiều công sức hơn cho quản lý sự phức tạp của hệ thống
- Kết thúc dự án có thể chưa phát hiện được các rủi ro tiềm ẩn
- Cần nhiều kinh nghiệm khi tiến hành

MÔ HÌNH XOẮN ỐC

- Kết hợp **tính lặp** của mô hình lặp và tính **hệ thống, dễ kiểm soát** của mô hình thác nước
- Phát hành các **sản phẩm tăng dần** một cách tuần tự theo các bước lặp
- Ý **kiến phản hồi** từ khách hàng/người dùng là cơ sở tiến hành các bước lặp tiếp theo
- Là mô hình được **sử dụng rộng rãi** trong phát triển phần mềm

MÔ HÌNH XOẮN ỐC



NGỮ CẢNH ÁP DỤNG

- Ràng buộc về ngân sách
- Đề cao phát hiện rủi ro
- Người dùng chưa chắc về yêu cầu của mình
- Yêu cầu phức tạp và cần sự đánh giá sớm, liên tục
- Tiếp nhận sớm những phản hồi từ khách hang
- Có thể chấp nhận những thay đổi lớn

ƯU ĐIỂM

- Thích nghi thay đổi yêu cầu tốt
- Cho phép sử dụng rộng rãi các bản mẫu
- Yêu cầu ngày càng được nắm bắt chính xác hơn
- Người dung có thể thấy được “hình hài” của sản phẩm sớm
- Hệ thống có thể được chia thành các phần con và các thành phần rủi ro có thể được xây dựng trước, đầu tư kĩ hơn

NHƯỢC ĐIỂM

- Công tác quản lý và quy trình khá phức tạp
- Khó xác định thời gian kết thúc dự án
- Không thích hợp dành cho dự án nhỏ, độ rủi ro thấp, và có thể tốn nhiều chi phí nếu quy mô dự án nhỏ
- Các giai đoạn giữa cần lập nhiều tài liệu

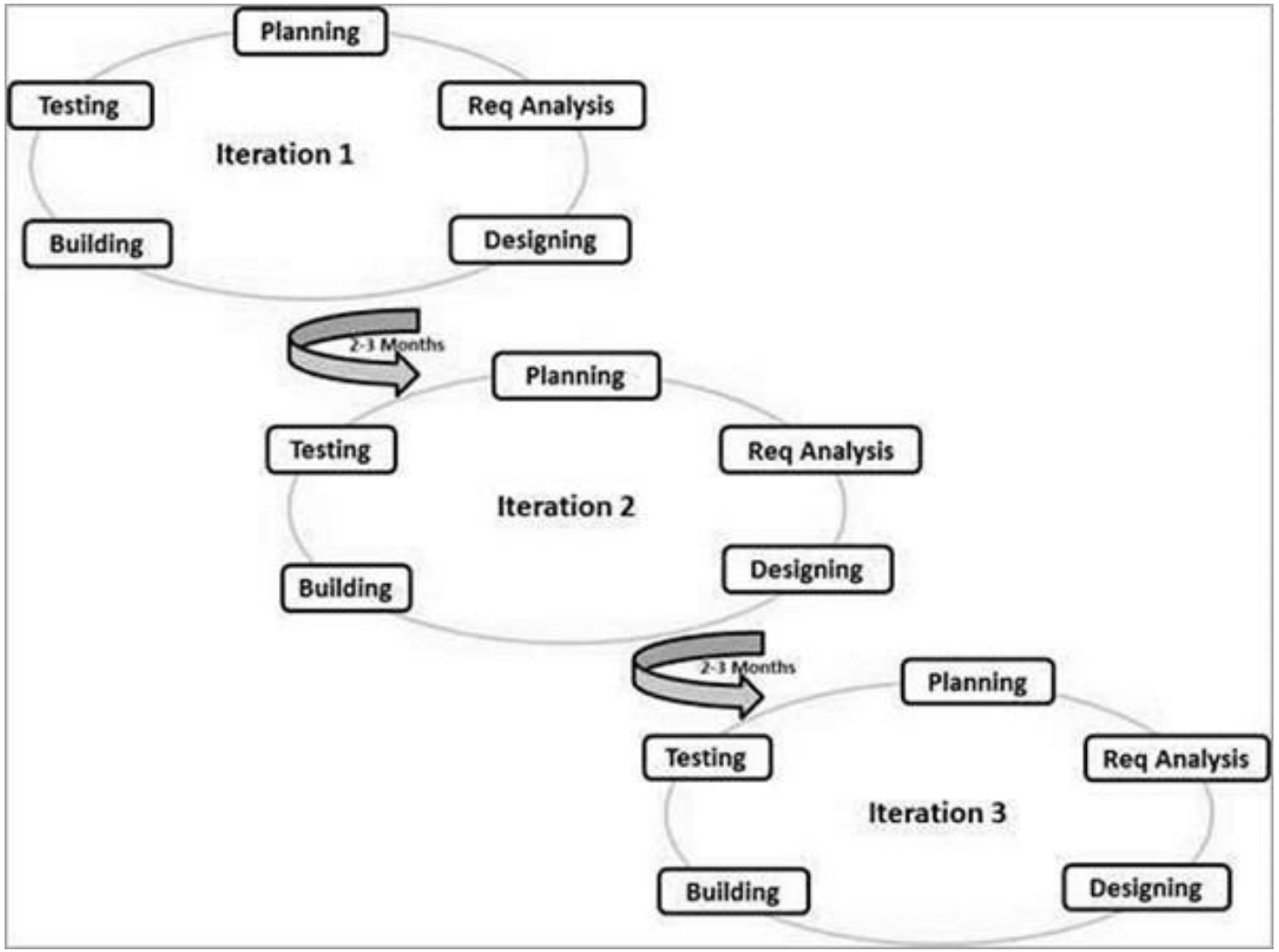
MÔ HÌNH AGILE

- Kết hợp giữa các mô hình quy trình **lặp** và **tăng dần**
- Tập trung vào khả năng **thích ứng quy trình** và sự **hài lòng của khách hàng** bằng cách cung cấp nhanh chóng mẫu sản phẩm
- Chia sản phẩm thành các **bản dựng** nhỏ tăng dần
- Mỗi lần lặp thường kéo dài từ khoảng 1-3 tuần
- Mỗi lần lặp lại liên quan đến các nhóm chức năng thuộc các pha khác nhau như: Planning, Requirements, Analysis, Design, Coding, Unit Testing, Acceptance Testing

MÔ HÌNH AGILE

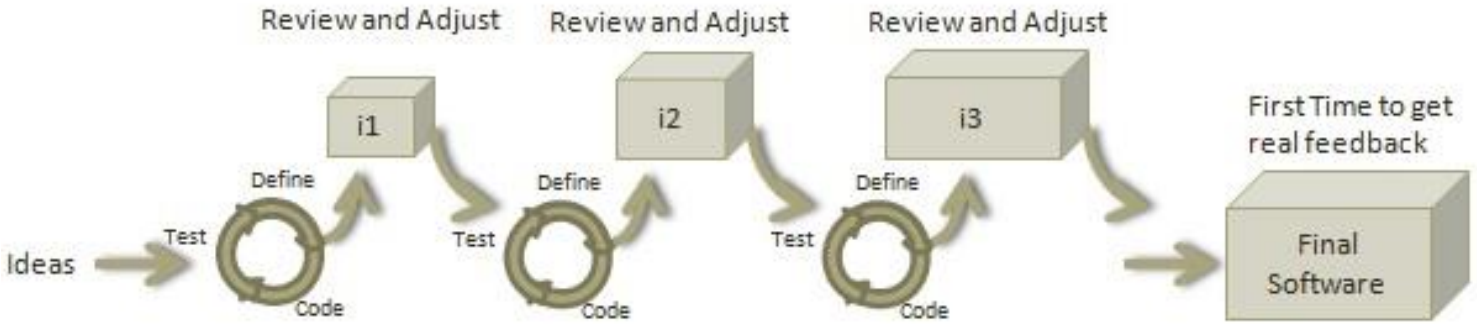
- Là phương pháp đề cao tính "linh hoạt" và hướng đến việc đáp ứng tốt yêu cầu
- Các tác vụ được chia thành các **hộp thời gian** (small time frames) để cung cấp các tính năng cụ thể cho một bản phát hành
- Mỗi bản dựng được **tăng dần về các tính năng**, bản dựng cuối cùng chứa tất cả các tính năng theo yêu cầu của khách hàng
- Là phương pháp ngày càng phổ biến do tính linh hoạt và khả năng thích ứng của nó

MÔ HÌNH AGILE



MÔ HÌNH AGILE

Traditional Method



Agile Method

MÔ HÌNH AGILE

- Các phương thức Agile phổ biến nhất bao gồm:
  - Rational Unified Process (1994)
  - Scrum (1995)
  - Crystal Clear
  - Extreme Lập trình (1996)
  - Phát triển phần mềm thích ứng (Adaptive Software Development)
  - Phát triển dựa trên tính năng
  - Phương pháp phát triển hệ thống động (DSDM) (1995)

MÔ HÌNH AGILE

- Các tuyên ngôn của Agile:
  - **Các cá nhân và tương tác:** Trong phát triển Agile, tự tổ chức và tạo động lực rất quan trọng, cũng như các tương tác như cùng vị trí và lập trình cặp
  - **Phần mềm làm việc:** Phần mềm hỗ trợ làm việc được coi là phương tiện giao tiếp tốt nhất với khách hàng để hiểu các yêu cầu của họ, thay vì chỉ phụ thuộc vào tài liệu
  - **Hợp tác với khách hàng:** Tương tác khách hàng liên tục là rất quan trọng để có được yêu cầu sản phẩm phù hợp
  - **Ứng phó với thay đổi:** Phát triển Agile tập trung vào các phản ứng nhanh với thay đổi và phát triển liên tục

CÁC ĐẶC TRƯNG CỦA AGILE

- Dựa trên các phương pháp phát triển **phần mềm thích ứng, thay đổi linh hoạt** (SDLC truyền thống như mô hình thác nước dựa trên phương pháp dự đoán)
- Sản phẩm được **kiểm tra rất thường xuyên**, thông qua các lần lặp lại phát hành, **giảm thiểu rủi ro** của bất kỳ thất bại lớn nào trong tương lai
- **Tương tác khách hàng** là xương sống của phương pháp Agile này và giao tiếp mở với tài liệu tối thiểu là các tính năng tiêu biểu của môi trường phát triển Agile
- Các nhóm nhanh nhẹn làm việc **phối hợp chặt chẽ** với nhau và thường được đặt ở cùng một vị trí địa lý

ƯU ĐIỂM

- Tiếp cận rất thực tế để phát triển phần mềm
- Thúc đẩy tinh thần đồng đội và đào tạo chéo
- Chức năng có thể được phát triển nhanh chóng
- Yêu cầu tài nguyên là tối thiểu
- Thích hợp cho các yêu cầu cố định hoặc thay đổi
- Hỗ trợ xây dựng giải pháp cục bộ

ƯU ĐIỂM

- Thích ứng tốt với môi trường thay đổi liên tục
- Sử dụng ít các quy định, tài liệ
- Cho phép phát triển đồng thời và phân phối trong bối cảnh kế hoạch tổng thể
- Ít hoặc không cần lập kế hoạch
- Dễ quản lý
- Cung cấp sự linh hoạt cho người lập trình

NHƯỢC ĐIỂM

- Không phù hợp để xử lý các phụ thuộc phức tạp
- Nhiều rủi ro về tính bền vững, khả năng duy trì và mở rộng
- Cần có một kế hoạch tổng thể, một nhà lãnh đạo nhanh nhẹn và kinh nghiệm
- Quản lý phân phối nghiêm ngặt chỉ ra phạm vi, chức năng sẽ được phân phối và điều chỉnh để đáp ứng thời hạn
- Phụ thuộc nhiều vào tương tác của khách hàng
- Sự phụ thuộc cá nhân rất cao
- Chuyển giao công nghệ cho các thành viên nhóm mới có thể khá khó khăn do thiếu tài liệu

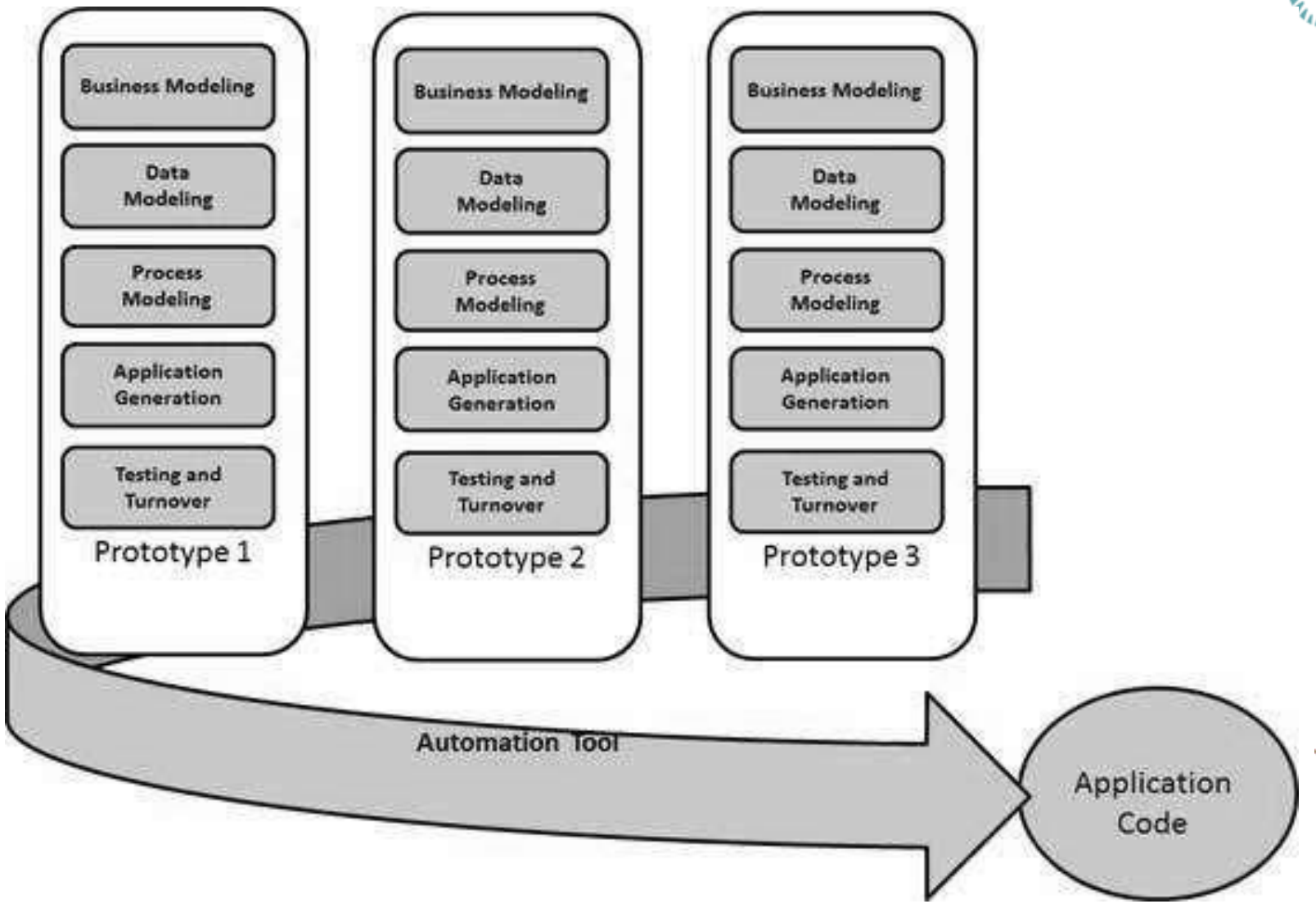
MÔ HÌNH RAD

- Mô hình RAD (Phát triển ứng dụng nhanh) dựa trên **nguyên mẫu** và **phát triển lặp** mà **không có kế hoạch cụ thể** liên quan
- Tập trung vào việc thu thập các yêu cầu của khách hàng thông qua các hội thảo hoặc các nhóm tập trung
- **Thử nghiệm sớm** các nguyên mẫu của khách hàng bằng cách sử dụng khái niệm lặp, tái sử dụng các nguyên mẫu (thành phần) hiện có, tích hợp liên tục và giao hàng nhanh

MÔ HÌNH RAD

- Là một phương pháp phát triển phần mềm đề cao việc **ít lập kế hoạch** và **tạo mẫu nhanh**
- Các **mô đun** chức năng được **phát triển song song** dưới dạng nguyên mẫu và được tích hợp để tạo ra sản phẩm hoàn chỉnh để phân phối sản phẩm nhanh hơn
- Vì không có kế hoạch trước chi tiết, nên việc kết hợp các thay đổi trong quá trình phát triển sẽ **linh hoạt** hơn
- Các nhóm nhỏ bao gồm các nhà phát triển, chuyên gia hệ thống, đại diện khách hàng và các tài nguyên CNTT khác làm việc liên tục trên thành phần hoặc nguyên mẫu
- Đảm bảo rằng các nguyên mẫu được phát triển có thể **tái sử dụng**

MÔ HÌNH RAD


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 3
Automation Tool
Application Code

MÔ HÌNH RAD

- **Business Modeling:** thiết kế theo luồng thông tin và phân phối thông tin giữa các kênh kinh doanh khác nhau, tìm thông tin quan trọng cho doanh nghiệp
- **Data Modeling:** Thông tin ở bước trên được xem xét và phân tích để tạo thành các bộ đối tượng dữ liệu quan trọng cho doanh nghiệp
- **Process Modeling:** Các bộ đối tượng dữ liệu được chuyển đổi để thiết lập luồng thông tin kinh doanh cần thiết để đạt được các mục tiêu kinh doanh cụ thể theo mô hình kinh doanh. Mô tả quy trình để thêm, xóa, truy xuất hoặc sửa đổi một đối tượng dữ liệu được đưa ra.

MÔ HÌNH RAD

- **Application Generation:** Hệ thống thực tế được xây dựng và mã hóa được thực hiện bằng cách sử dụng các công cụ tự động hóa để chuyển đổi các mô hình quá trình và dữ liệu thành các nguyên mẫu thực tế
- **Testing and Turnover:** Thời gian thử nghiệm tổng thể được giảm trong mô hình RAD vì các nguyên mẫu được kiểm tra độc lập trong mỗi lần lặp. Tuy nhiên, luồng dữ liệu và giao diện giữa tất cả các thành phần cần phải được kiểm tra kỹ lưỡng với phạm vi kiểm tra hoàn chỉnh

NGỮ CẢNH SỬ DỤNG

- Hệ thống có thể được **mô đun hóa** để được phân phối theo cách tăng dần
- **Năng lực** đưa ra mô hình tốt
- **Ngân sách** cho phép sử dụng các công cụ tạo mã tự động
- Các **chuyên gia hệ thống** có sẵn kiến thức kinh doanh có liên quan
- Các **yêu cầu thay đổi** trong dự án và các nguyên mẫu làm việc sẽ được trình bày cho khách hàng mỗi 2-3 tháng

ƯU ĐIỂM

- Thích nghi với việc thay đổi yêu cầu
- Tiến độ có thể được đo lường
- Thời gian lặp thể ngắn
- Hiệu quả trong khi cần ít nhân lực trong thời gian ngắn
- Giảm thời gian phát triển
- Tăng khả năng tái sử dụng của các thành phần
- Đánh giá ban đầu được diễn ra nhanh chóng
- Khuyến khích phản hồi của khách hàng
- Giải quyết rất nhiều vấn đề tích hợp

NHƯỢC ĐIỂM

- Phụ thuộc vào đội ngũ mạnh về kỹ thuật xác định yêu cầu
- Chỉ dung cho hệ thống có thể được mô đun hóa
- Yêu cầu các nhà phát triển / thiết kế có tay nghề cao
- Phụ thuộc cao vào kỹ năng xây dựng mẫu
- Không thể áp dụng cho các dự án có chi phí thấp
- Quản lý phức tạp hơn
- Thích hợp cho các hệ thống dựa trên thành phần và có thể mở rộng
- Yêu cầu sự tham gia xuyên suốt của người dùng
- Thích hợp cho dự án đòi hỏi thời gian phát triển ngắn hơn

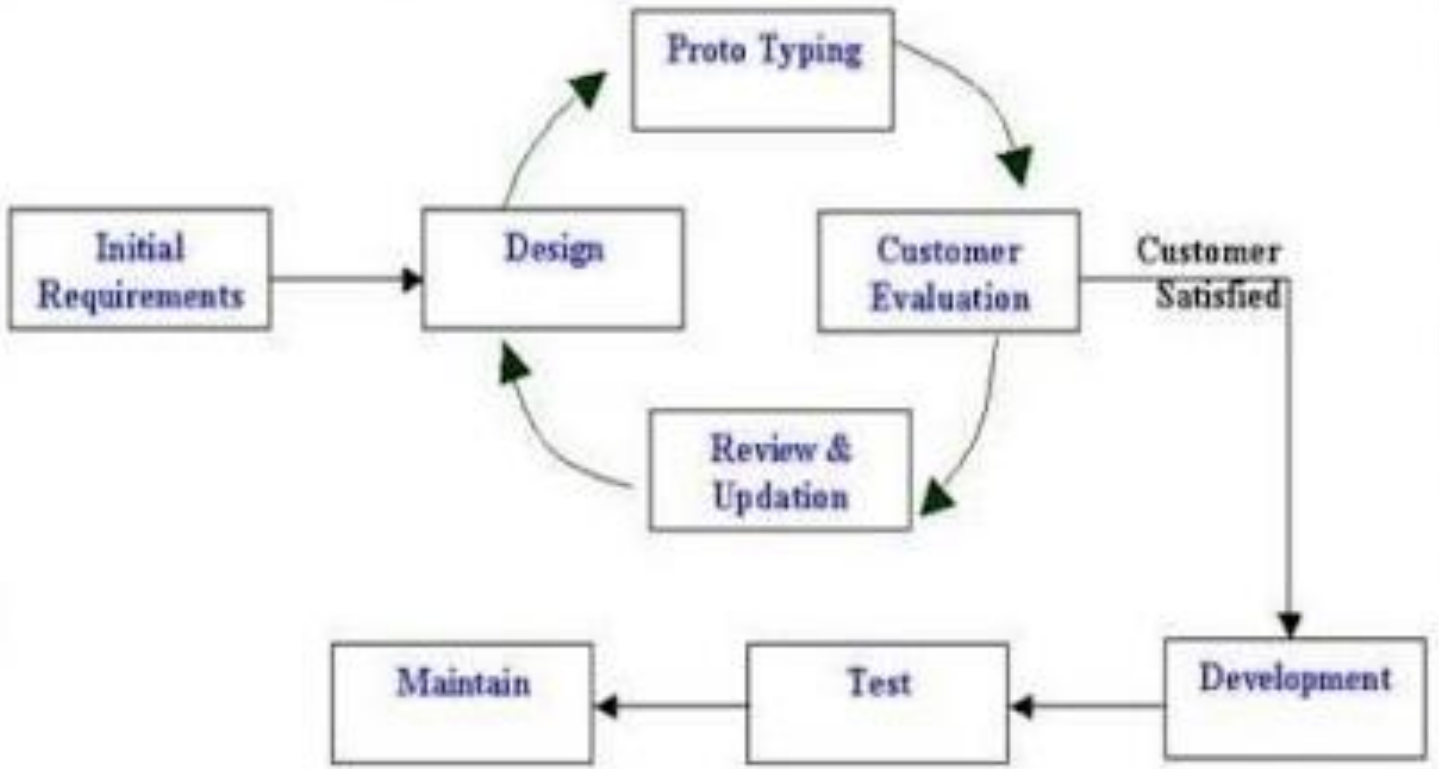
MÔ HÌNH PROTOTYPE

- Xây dựng trên các **nguyên mẫu** ứng dụng phần mềm
- Nguyên mẫu hiển thị **chức năng** của sản phẩm đang được phát triển, nhưng có thể không thực sự giữ logic chính xác của phần mềm gốc
- Phương pháp này phổ biến vì nó cho phép **thu hút và giúp hiểu các yêu cầu của khách hàng** ở giai đoạn phát triển ban đầu
- Prototyping được sử dụng để cho phép người dùng **đánh giá các đề xuất** của nhà phát triển và thử chúng trước khi thực hiện

CÁC BƯỚC KHI TẠO PROTOTYPE

- **Basic Requirement Identification**
  - Nắm các yêu cầu sản phẩm rất cơ bản, đặc biệt là về giao diện người dùng
  - Hiệu suất và bảo mật có thể bị bỏ qua
- **Developing the initial Prototype**
  - Đáp ứng các yêu cầu rất cơ bản và giao diện người
  - Mang lại cái nhìn và cảm nhận tương tự cho khách hàng
- **Review of the Prototype:** trình bày cho khách hàng và lấy các nhận xét và yêu cầu mới
- **Revise and Enhance the Prototype**
  - Thảo luận và đàm phán với khách hang về việc cải tiến
  - Chú ý sử dụng nguyên mẫu ngang (giao diện) hoặc nguyên mẫu dọc (chức năng và xử lý)

MÔ HÌNH PROTOTYPE



CÁC LOẠI NGUYÊN MẪU

- **Throwaway/Rapid Prototyping**
  - Phân tích yêu cầu tối thiểu để xây dựng nguyên mẫu
  - Khi các yêu cầu thực tế được hiểu rõ hơn, nguyên mẫu sẽ bị loại bỏ và hệ thống thực tế được phát triển
- **Evolutionary Prototyping**
  - Xây dựng các nguyên mẫu có chức năng thực tế (chức năng chính)
  - Nguyên mẫu được phát triển tạo thành core của hệ thống thực tế
- **Incremental Prototyping**
  - Xây dựng nhiều nguyên mẫu chức năng con
  - Tích hợp tất cả các nguyên mẫu có sẵn để tạo thành một hệ thống hoàn chỉnh

CÁC LOẠI NGUYÊN MẪU

Extreme Prototyping

- Sử dụng trong lĩnh vực phát triển web
- Gồm ba giai đoạn liên tiếp:
  - Tạo nguyên mẫu cơ bản với tất cả các trang hiện có được trình bày ở định dạng HTML
  - Xử lý dữ liệu được mô phỏng bằng lớp dịch vụ nguyên mẫu
  - Các dịch vụ được triển khai và tích hợp vào nguyên mẫu cuối cùng
- Thu hút sự chú ý đến giai đoạn thứ hai của quy trình (Developing the initial Prototype), trong đó một UI đầy đủ chức năng được phát triển mà rất ít liên quan đến các dịch vụ thực tế

NGỮ CẢNH SỬ DỤNG

- Phát triển các hệ thống có mức độ tương tác cao (như hệ thống trực tuyến)
- Cần người dùng điền vào biểu mẫu hoặc qua các màn hình khác nhau trước khi dữ liệu được xử lý có thể sử dụng tạo mẫu

ƯU ĐIỂM

- Quyết định phải được thực hiện rất cẩn thận
- Tăng sự tham gia và tính tương tác của người dùng
- Người dùng hiểu rõ hơn về hệ thống đang xây dựng
- Giảm thời gian và chi phí vì các rủi ro có thể được phát hiện sớm
- Phản hồi của người dùng nhanh
- Các chức năng còn thiếu có thể được xác định dễ dàng
- Dễ xác định và giải quyết các khó khăn

NHƯỢC ĐIỂM

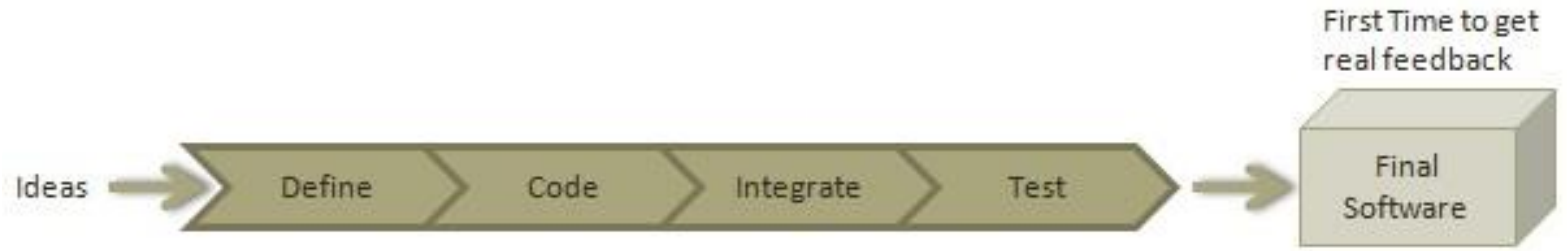
- Yêu cầu không đủ do phụ thuộc quá nhiều vào nguyên mẫu
- Người dùng có thể bị lẫn lộn trong các nguyên mẫu và hệ thống thực tế
- Các nhà phát triển có thể cố gắng sử dụng lại các nguyên mẫu hiện có để xây dựng hệ thống thực tế, ngay cả khi nó không khả thi về mặt kỹ thuật
- Nỗ lực đầu tư vào việc xây dựng các nguyên mẫu có thể là tốn kém nếu không được giám sát đúng cách

PHƯƠNG PHÁP AGILE

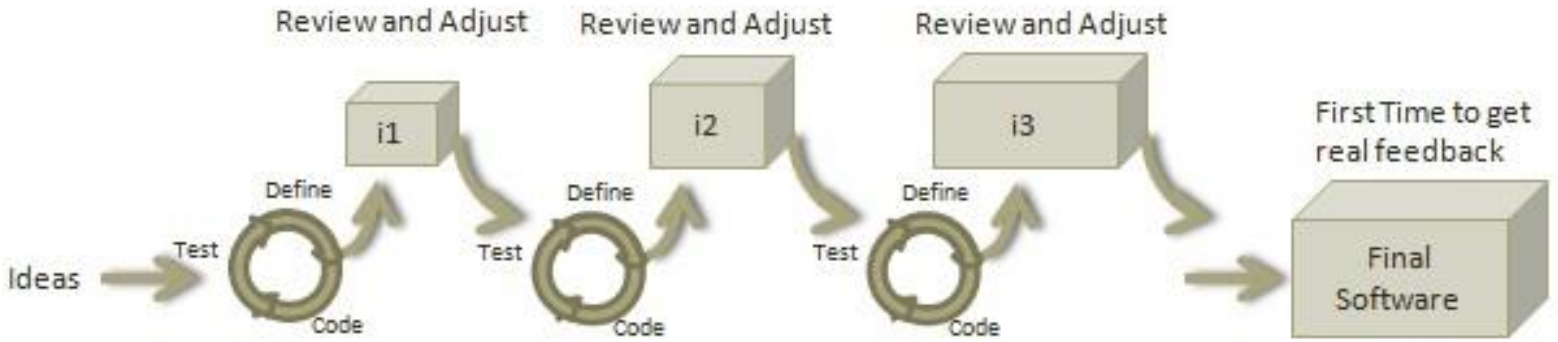
Giới thiệu

- Là một phương pháp phát triển phần mềm để xây dựng phần mềm tăng dần
- Sử dụng các vòng lặp ngắn từ 1 đến 4 tuần để quá trình phát triển phù hợp với nhu cầu kinh doanh thay đổi
- Áp dụng quy trình phản hồi thường xuyên trong đó một sản phẩm khả thi được phân phối sau 1 đến 4 tuần lặp lại

Giới thiệu



Traditional Method



Agile Method

Các vai trò

Scrum Master

- Là trưởng nhóm và người hỗ trợ, giúp các thành viên trong nhóm tuân theo các thực hành nhanh nhẹn để họ có thể đáp ứng các cam kết của họ
- Trách nhiệm gồm:
  - Đảm bảo hợp tác chặt chẽ giữa tất cả các vai trò và chức năng
  - Tháo gở các khó khăn, tránh xáo trộn
  - Theo dõi tiến trình công việc
  - Để đảm bảo các quy trình kiểm tra và thích ứng của Agile được tận dụng đúng cách: các cuộc họp hàng ngày, các cuộc họp theo kế hoạch, demo, đánh giá, cuộc họp hồi hướng và tạo điều kiện nhóm làm việc và ra quyết định dễ dàng

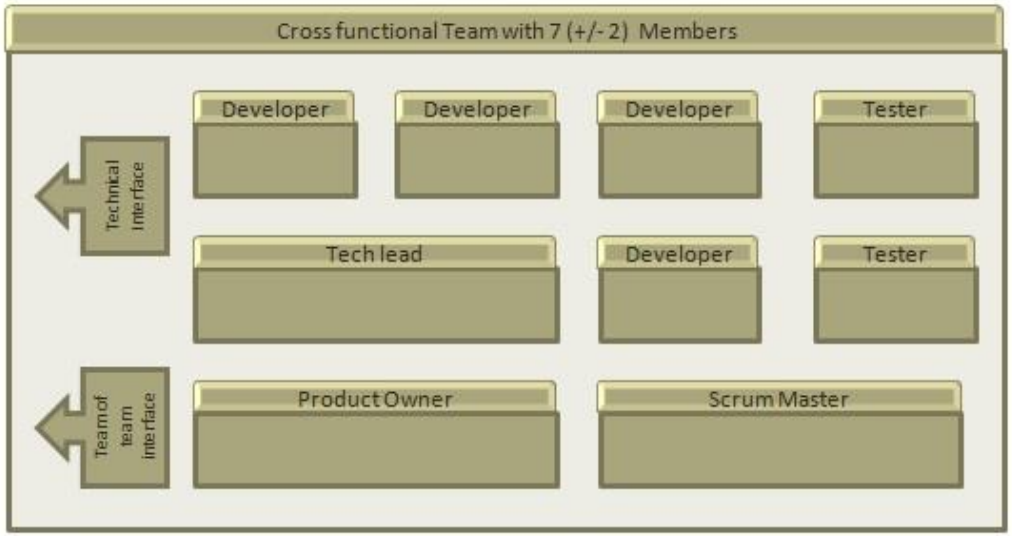
Các vai trò

Product Owner

- Là người điều khiển sản phẩm từ khía cạnh kinh doanh
- Trách nhiệm gồm:
  - Xác định các yêu cầu và xác định độ ưu tiên
  - Xác định nội dung và ngày phát hành
  - Có một vai trò tích cực trong lập kế hoạch mang tính lặp lại và đề xuất các cuộc họp trong kế hoạch
  - Đảm bảo rằng nhóm đang làm việc đúng yêu cầu
  - Đại diện cho tiếng nói của khách hàng
  - Xác định các tiêu chuẩn về sản phẩm

Đội ngũ liên quan

- Mỗi team gồm 5 đến 9 thành viên
- Thông thường, một nhóm gồm 3 đến 4 nhà phát triển: 1 người thử nghiệm, 1 lãnh đạo kỹ thuật, 1 chủ sở hữu sản phẩm và 1 chủ scrum
- Product Owner và Scrum master được coi là một phần của Team Interface, trong khi các thành viên khác là một phần của Technical Interface



Lập kế hoạch công việc

- Các nhóm làm việc trong các lần lặp để phân phối các câu chuyện/yêu cầu của người dùng trong đó mỗi lần lặp là từ 10 đến 15 ngày
- Mỗi câu chuyện của người dùng được lên kế hoạch dựa trên mức độ ưu tiên và kích thước của nó
- Dựa vào năng lực, nhân lực của nhóm để quyết định phạm vi công việc sẽ thực hiện tiếp theo

Các khái niệm liên quan

- Point
  - Kết quả của một nhóm có thể cam kết
  - Một điểm có thời lượng khoảng 8 giờ (1 ngày)
  - Mỗi câu chuyện được ước tính bằng điểm
- Capacity
  - Kết quả một cá nhân có thể cam kết
  - Được ước tính bằng giờ

Các khái niệm liên quan

User Story

- Là một yêu cầu của người dung về chức năng
- Có hai dạng:
  - Với <Vai trò …> tôi muốn <chức năng…> nhằm mang lại <giá trị doanh nghiệp….>
  - Để đạt được <giá trị doanh nghiệp…> với <vai trò….> tôi muốn <chức năng…>
- Trong quá trình lập kế hoạch, một ước tính sơ bộ được đưa ra cho câu chuyện của người dùng sử dụng đo bằng các Point. Trong quá trình lập kế hoạch lặp lại, câu chuyện được chia thành các nhiệm vụ
- Các câu chuyện sẽ được phân thành các stask

Các khái niệm liên quan

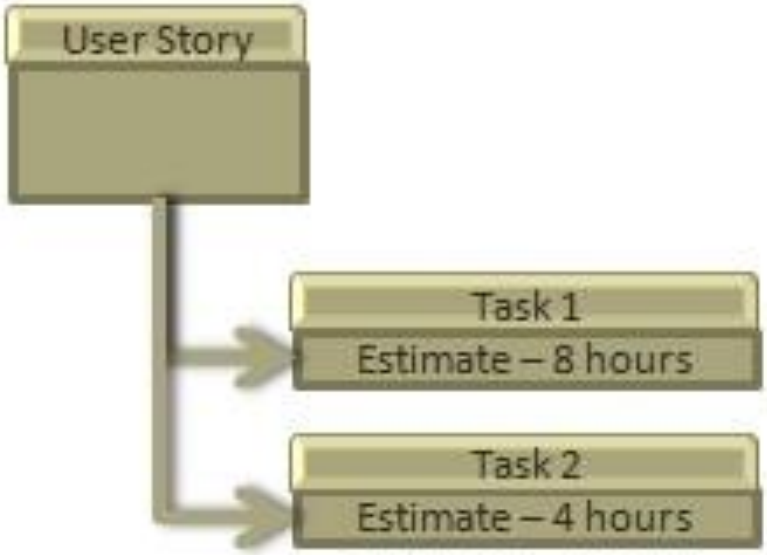
Iteration

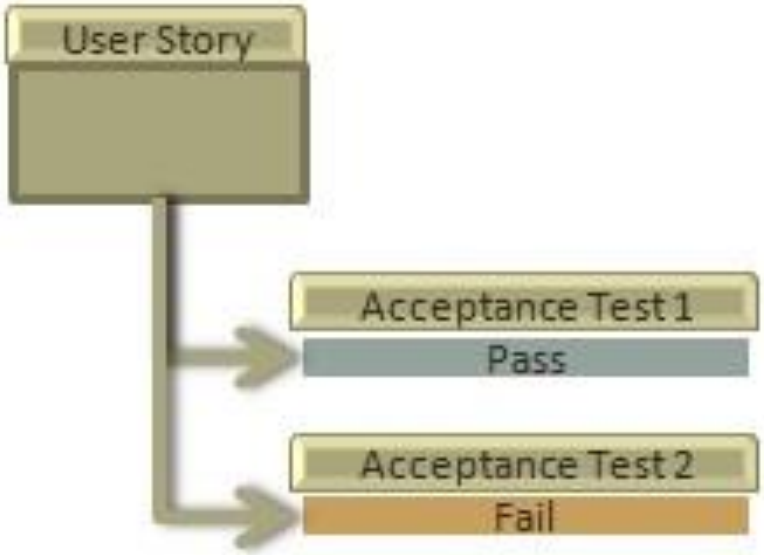
- Một tập hợp các câu chuyện được xử lý theo thời gian và được chấp nhận trong quá trình phát hành sản phẩm
- Được xác định trong cuộc họp lập kế hoạch và hoàn thành trong một cuộc họp đánh giá và thử nghiệm
- Còn được gọi là **sprint**

Quan hệ giữa User Stories và Tasks

- Câu chuyện của người dùng liên quan đến công việc sẽ được thực hiện, nó định nghĩa những gì người dùng cần
- Nhiệm vụ nói về cách hoàn thành công việc. Nó định nghĩa cách một chức năng được thực hiện
- Câu chuyện được thực hiện bởi các nhiệm vụ. Mỗi câu chuyện là một tập hợp các nhiệm vụ
- Câu chuyện được chia thành các nhiệm vụ trong kế hoạch
- Nhiệm vụ được ước tính theo giờ, thường từ 2 đến 12 giờ
- Câu chuyện được xác nhận bằng cách sử dụng acceptance tests

Quan hệ giữa User Stories và Tasks





Khi nào câu chuyện được hoàn thành?

- Thỏa mãn các tiêu chí sau:
  - Tất cả các task được hoàn thành
  - Tất cả các acceptance tests được thông qua
  - Không có thêm lỗi, công việc phát sinh
  - Product owner chấp nhận
  - Phát hành đến người dùng cuối

12 tuyên bố của Agile

1. Customer Satisfaction
2. Welcome Change
3. Deliver a Working Software
4. Collaboration
5. Motivation
6. Face-to-face Conversation
7. Measure the Progress as per the Working Software
8. Maintain Constant Pace
9. Monitoring
10. Simplicity
11. Self-organized Teams
12. Review the Work Regularly

Chi tiết: ***https://agilemanifesto.org/***

Đặc điểm phương pháp Agile

1. Lặp/tăng và sẵn sàng cả tiến
2. Giao tiếp face-to-face
3. Vòng lặp phản hồi thông tin

Lặp/tăng và sẵn sàng cả tiến

- Không có kế hoạch dài hạn
- Các lần lặp được lên kế hoạch trong khoảng thời gian ngắn khác nhau, ví dụ: 1 đến 4 tuần
- Sau khi demo, đánh giá nhận xét được thực hiện và được lên kế hoạch để được tích hợp vào phần mềm làm việc theo yêu cầu

Giao tiếp face-to-face

- Mỗi nhóm nên có một đại diện khách hàng làm product owner
- Đại diện này được ủy quyền hành động thay mặt cho các bên liên quan và anh ta có thể trả lời các truy vấn của nhà phát triển ở giữa các lần lặp
- Một bảng thể hiện kết quả thường được đặt nổi tại nơi làm việc, tóm tắt cập nhật về tình trạng của một dự án

Vòng lặp phản hồi thông tin

- Báo cáo/phát biểu hàng ngày là một văn hóa tích cực cho hoạt động nhóm
- Đây là một phiên họp ngắn, trong đó mỗi thành viên trong nhóm báo cáo với nhau về tình trạng của những gì họ đã làm, phải làm gì tiếp theo và bất kỳ vấn đề nào họ đang phải đối mặt

Hoạt động báo cáo hằng ngày

- Là một cuộc họp hàng ngày giữa tất cả các thành viên nhằm nắm bắt nhanh hiện trạng
- Giúp cập nhật thường xuyên thực trạng và giúp các thành viên trong nhóm tập trung giải quyết nhanh hơn
- Là một việc phải làm, bất kể năng lực đội ngũ như thế nào, hoạt động ở đâu

Hoạt động báo cáo hằng ngày

- Tổ chức trong khoảng **15 phút**
- Mỗi thành viên phải trả lời **3 câu hỏi** quan trọng
  - Hôm qua đã làm gì?
  - Hôm nay sẽ làm gì?
  - Trở ngại nào gặp phải? Lý do là?
- Để cập nhật và thông báo trạng thái, không phải nhằm mục đích thảo luận
- Để thảo luận, các thành viên trong nhóm nên sắp xếp một cuộc họp khác vào một thời điểm khác
- Những người tham gia thường đứng lên trình bày để cuộc họp kết thúc nhanh chóng

Hoạt động báo cáo hằng ngày

Tại sao nên báo cáo hằng ngày

- Nhóm có thể đánh giá tiến trình hàng ngày và xem liệu họ có thể đảm bảo kết quả theo kế hoạch
- Mỗi thành viên trong nhóm thông báo tất cả về các cam kết của mình trong ngày
- Cung cấp khả năng thấu hiểu xuyên suốt trong nhóm hoặc cho bất kỳ sự chậm trễ hoặc trở ngại nào

Người tham gia

- Scrum master, product owner và nhóm phát triển
- Có thểm bao gồm các bên liên quan và khách hàng
- Trách nhiệm của chủ Scrum master là ghi lại các trao đổi của từng thành viên và các vấn đề họ đang gặp phải

Hoạt động báo cáo hằng ngày

- **Khác nhau vị trí địa lý và múi giờ**
  - Chọn thành viên trên cơ sở luân phiên, người có thể tham dự cuộc họp độc lập của các đội nằm ở các múi giờ khác nhau
  - Mỗi nhóm có hoạt động báo cáo hằng ngày riêng, cập nhật trạng thái của của hoạt động báo cáo bằng các công cụ như Rally, SharePoint, Wikis,…
  - Có nhiều công cụ giao tiếp sẵn sàng như cuộc gọi hội nghị, hội nghị video, tin nhắn tức thời hoặc bất kỳ công cụ chia sẻ nào

Tính hoàn thành

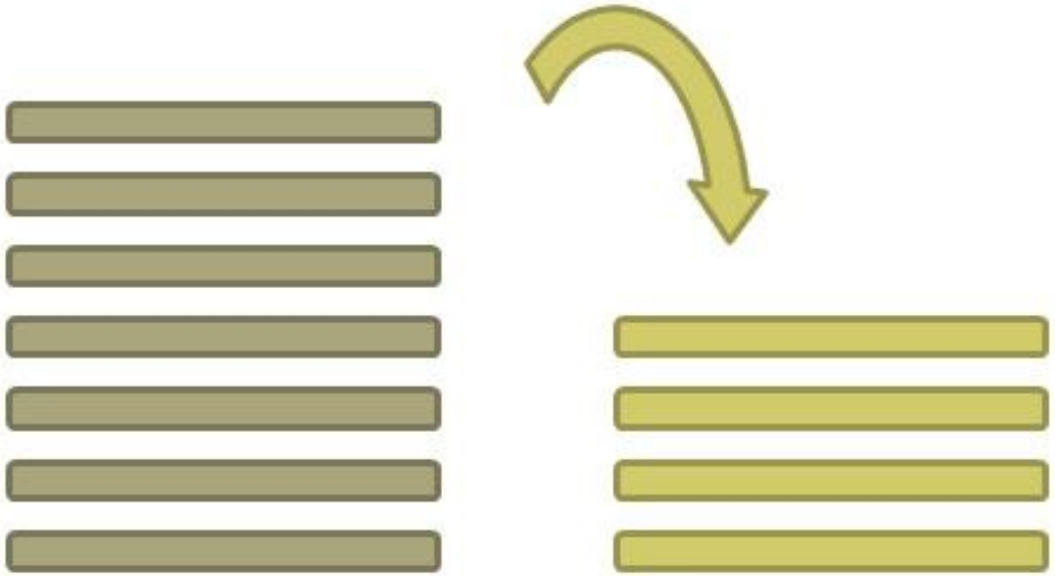
- **User Story**
  - Tất cả các mã liên quan đã được cập nhật
  - Các unit test và acceptance test đã được thông qua
  - Văn bản trợ giúp (help text) được xây dựng
  - Product owner đã đồng ý
- **Iteration**
  - Sao lưu sản phẩm đã hoàn tất
  - Hiệu suất đã được thử nghiệm
  - Câu chuyện của người dùng đã được chấp nhận hoặc chuyển sang lần lặp tiếp theo
  - Các sai sót đã được sửa chữa hoặc hoãn lại cho lần lặp tiếp theo

Tính hoàn thành

- **Release**
  - Hệ thống được kiểm tra tổng thể
  - Hiệu suất được điều chỉnh
  - Xác nhận an ninh được thực hiện
  - Kế hoạch khắc phục thảm họa được thử nghiệm

Kế hoạch release sản phẩm

- Tạo một kế hoạch để cung cấp một gia tăng cho sản phẩm
- Được thực hiện sau mỗi 2 đến 3 tháng



Kế hoạch release sản phẩm

Người liên quan

- Scrum Master
- Product Owner
- Agile Team
- Stakeholders
  - Customers
  - Program managers
  - Subject matter experts

Kế hoạch release sản phẩm

❖ Yêu cầu

- Product backlog được xếp hạng (được quản lý bởi product owner)
- Thông thường có 5-10 tính năng được product owner đưa vào trong bản phát hành
- Khả năng, tốc độ hoàn thành công việc của nhóm
- Tầm nhìn tốt
- Mục tiêu kinh doanh và mục tiêu nghiệp vụ
- Xác nhận xem có cần product backlog mới không

Kế hoạch release sản phẩm

Materials Required

- Nơi thông báo nội dung, mục đích
- Flip charts, bảng trắng, bút,..
- Máy chiếu, cách chia sẻ máy tính,dữ liệu, công cụ cần thiết
- Dữ liệu cần thiết cho lập kế hoạch

Kế hoạch release sản phẩm

Planning Data

- Bước lặp trước đó, kết quả của kế hoạch phát hành
- Phản hồi từ các bên liên quan khác nhau về sản phẩm, thị trường và deadline
- Kế hoạch của các phiên bản/lần lặp trước
- Tính năng hoặc lỗi có liên quan
- Tốc độ từ các bản phát hành/ước tính trước đó.
- Lịch làm việc của tổ chức và cá nhân
- Dữ liệu từ các nhóm khác và các chuyên gia liên quan

Kế hoạch release sản phẩm

Output

- Kế hoạch release
- Các cam kết
- Các vấn đề, mối quan tâm, sự phụ thuộc và giả định sẽ được theo dõi
- Các đề xuất để cải thiện kế hoạch release

Kế hoạch release sản phẩm

Các nội dung cần quan tâm

- Opening ceremony
- Product Vision, Roadmap
- Review previous releases
- Release name / theme
- Velocity
- Release schedule
- Issues and concerns
- Review and Update the Definition of Done
- Stories and items to be considered
- Determine sizing values

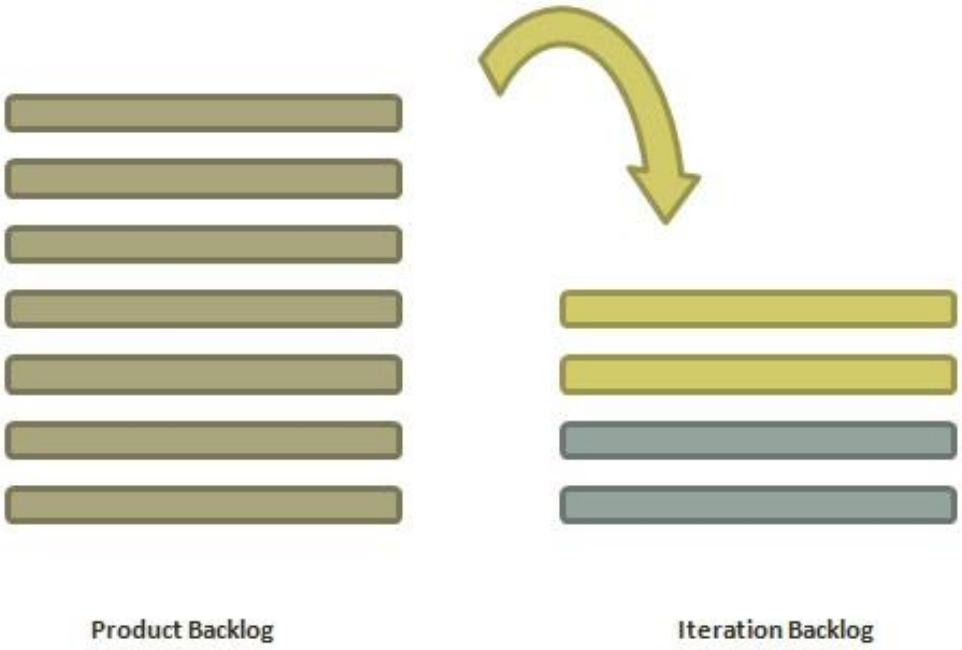
Kế hoạch release sản phẩm

Các nội dung cần quan tâm

- Coarse the size of stories
- Map stories to iterations
- New concerns or issues
- Dependencies and assumptions
- Commit
- Communication and logistics planning
- Parking lot
- Distribute Action items and action plans
- Retrospect
- Close

Kế hoạch cho Iteration

- Mục đích là để xây dựng bộ các mục trong product backlog
- Cam kết này được tính theo thời gian dựa trên độ dài của lần lặp và tốc độ làm việc của nhóm



Kế hoạch cho Iteration

- Người liên quan
  - Scrum Master
  - Product Owner
  - Agile Team

Kế hoạch cho Iteration

❖ Yêu cầu

- Các mục trong product backlog được xác định kích thước và các point liên quan đến story được chỉ ra
- Các danh mục phát triển được sắp xếp thứ tự từ product owner
- Các tiêu chí cho từng danh mục phát triển được xác định rõ

Kế hoạch cho Iteration

Quy trình

- Xác định các câu chuyện liên quan
- Chia những câu chuyện này thành các task và giao nhiệm vụ
- Mỗi nhiệm vụ được ước tính trong vài giờ
- Các ước tính này giúp các thành viên trong nhóm kiểm tra xem mỗi thành viên có bao nhiêu giờ nhiệm vụ cho lần lặp
- Các thành viên trong nhóm được phân công nhiệm vụ dựa trên năng lực và mức độ hoàn thành công việc

Kế hoạch cho Iteration

Tính toán tốc độ

- Đựa trên các lần lặp lại trong quá khứ
- Vận tốc là số đơn vị trung bình cần thiết để hoàn thành story trong một lần lặp
- Ví dụ: nếu đội có số point là 12, 14, 10 trong mỗi lần lặp cho ba lần lặp trước đó, thì nhóm có thể lấy 12 point là tốc độ cho lần lặp tiếp theo
- Dựa vào tốc độ có thể biết có bao nhiêu câu chuyện người dùng có thể được hoàn thành trong lần lặp hiện tại
- Nếu tốc độ nhanh thì có thể kéo thêm nhiều câu chuyện của người dùng vào, ngược lại các câu chuyện cũng có thể được chuyển sang lần lặp tiếp theo.

Kế hoạch cho Iteration

- **Task Capacity:** dựa trên 3 yếu tố
  - Số giờ làm việc lý tưởng trong một ngày
  - Số ngày có sẵn của mỗi người trong phần lặp
  - Lượng thời gian mà một thành viên dành riêng cho nhóm
  - Giả sử một nhóm có 5 thành viên, cam kết làm việc toàn thời gian (8 giờ một ngày) trong một dự án và không ai được nghỉ trong một lần lặp, thì khả năng nhiệm vụ cho một lần lặp hai tuần sẽ là:

$$5 \times 8 \times 10 = 400 \text{ giờ}$$

Kế hoạch cho Iteration

Các bước lập kế hoạch

- Product owner mô tả các mục được được xếp hạng cao nhất trong product backlog
- Nhóm mô tả các nhiệm vụ cần thiết để hoàn thành danh mục
- Các thành viên trong nhóm nhận các nhiệm vụ
- Các thành viên trong nhóm ước tính thời gian để hoàn thành mỗi nhiệm vụ
- Các bước này được lặp lại cho tất cả các mục trong vòng lặp
- Nếu bất kỳ cá nhân nào bị quá tải với các nhiệm vụ, thì nhiệm vụ của họ được phân phối giữa các thành viên khác trong nhóm

Product Backlog

- Là một danh sách các việc sẽ được thực hiện
- Các mục được xếp hạng với mô tả tính năng
- Các mục nên được chia thành các câu chuyện của người dùng

Product Backlog

Tại sao Product Backlog quan trọng

- Hỗ trợ các ước lượng cần thiết cho các tính năng
- Giúp lập kế hoạch lộ trình cho sản phẩm
- Giúp xếp hạng lại các tính năng
- Giúp xác định danh mục ưu tiên, team xếp hạng các mục và tiến hành công việc

Product Backlog

❖ Đặc điểm của Product Backlog

- Mỗi sản phẩm nên có Product Backlog gồm một tập lớn hoặc rất lớn các tính năng
- Nhiều nhóm có thể làm việc trên một Product Backlog duy nhất
- Xếp hạng các tính năng được thực hiện dựa trên giá trị kinh doanh, giá trị kỹ thuật, quản lý rủi ro hoặc theo chiến lược
- Các mục xếp hạng cao nhất được phân tách thành các stoty nhỏ hơn trong kế hoạch phát hành để có thể được hoàn thành trong các lần lặp

BẢNG TỪ VỰNG AGILE

KHÁI NIỆM	MÔ TẢ
Acceptance Criteria	Các tiêu chí được đặt ra bởi product owner/người dung dành cho tính năng
Backlog Grooming	Một quá trình liên tục trong đó product owner quản lý product backlog bằng cách nhận phản hồi từ agila team
Capacity	khối lượng công việc mà nhóm có thể thực hiện trong một lần lặp
Feature	Một cải tiến được thực hiện đối với một sản phẩm hoặc khả năng của giá trị đối với các bên liên quan
Iteration	Một mục công việc dựa trên chủ đề có thể được hoàn thành trong một hộp thời gian và được chấp nhận trong khi phát hành sản phẩm. Còn được gọi là Sprint
Increment	Là trạng thái thay đổi của sản phẩm khi nó trải qua quá trình phát triển dần dần. Nó thường được biểu diễn bằng các mốc hoặc số lần lặp cố định
Product Owner	Là thành viên của nhóm Agile delivery, truyền đạt những gì sẽ được thực hiện trong một bản phát hành/lặp lại, bảo vệ nhóm khỏi mọi thay đổi trong yêu cầu

BẢNG TỪ VỰNG AGILE

KHÁI NIỆM	MÔ TẢ
Product Backlog	Tập các cầu chức năng và phi chức năng
Product Backlog Items	Có thể là câu chuyện của người dùng, lỗi, các tính năng
Points	Đơn vị phổ biến được sử dụng để đo câu chuyện người dùng, tính năng hoặc bất kỳ mục danh mục đầu tư nào
Release	Một hộp thời gian nơi công việc được thực hiện để hỗ trợ phân phối mức tăng có thể kiểm tra được cho một phần mềm. Gồm nhiều lần lặp
Sprint	Tương tự Iteration
Timebox	Một khoảng thời gian cố định trong đó một sản phẩm có thể giao được sẽ được phát triển
Task	Đơn vị công việc góp phần hoàn thành câu chuyện của người dùng trong một lần lặp
User Story	Một tiêu đánh giá của người dùng
Velocity	Thước đo để cân trọng lượng công việc được chấp nhận trong một lần lặp hoặc bảng thời gian. Thông thường nó là tổng số point trong một lần lặp