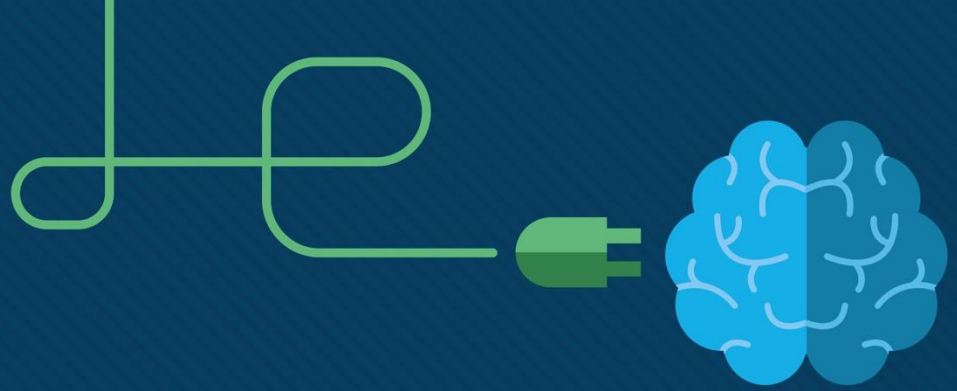
Trường ĐH Công nghệ Thông tin - ĐHQG TP. HCM

NT521 - Lập trình an toàn & Khai thác lỗ hổng phần mềm



Qui trình thiết kế và phát triển phần mềm (Phần 1)

DevNet Associate v1.0

Nội dung chính

- Phát triển phần mềm và quy trình phát triển phần mềm
- Mẫu thiết kế phần mềm – Design pattern
- Hệ thống quản lý phiên bản – Version control System
- Nền tảng lập trình
- Đánh giá và kiểm thử mã nguồn
- Định dạng dữ liệu

Phát triển phần mềm và Quy trình phát triển phần mềm

Thiết kế và phát triển phần mềm: Giới thiệu

- Qui trình phát triển phần mềm thường được gọi là: ”**S**oftware **D**evelopment **L**ife **C**ycle" (SDLC).
- SDLC không chỉ bao gồm việc lập trình, mà còn bao gồm: thu thập yêu cầu (gathering requirement), thiết kế (creating a proof of concept), kiểm thử (testing), và gỡ lỗi (fixing bugs).

Software Development Life Cycle (SDLC)

- **SDLC** là một qui trình phát triển phần mềm:
  - Gồm **6 giai đoạn**. Bắt đầu từ "**lên ý tưởng**" - "**phân phối phần mềm**".
  - Mỗi giai đoạn có đầu vào là kết quả của giai đoạn ngay trước nó.
- **Mô hình:**
  - Truyền thống: mô hình Thác nước (**Waterfall**)
  - Các phương pháp phát triển phần mềm linh hoạt được gọi chung là "**Agile Development**".

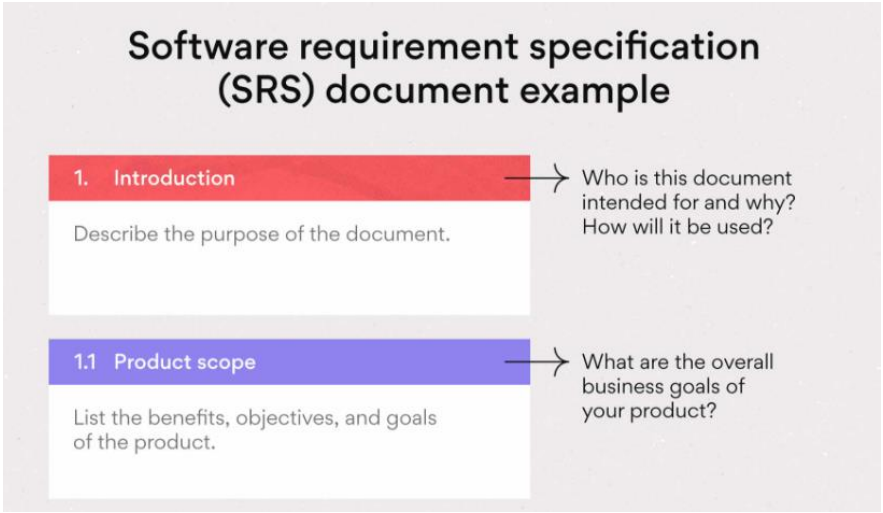
SDLC: Các giai đoạn

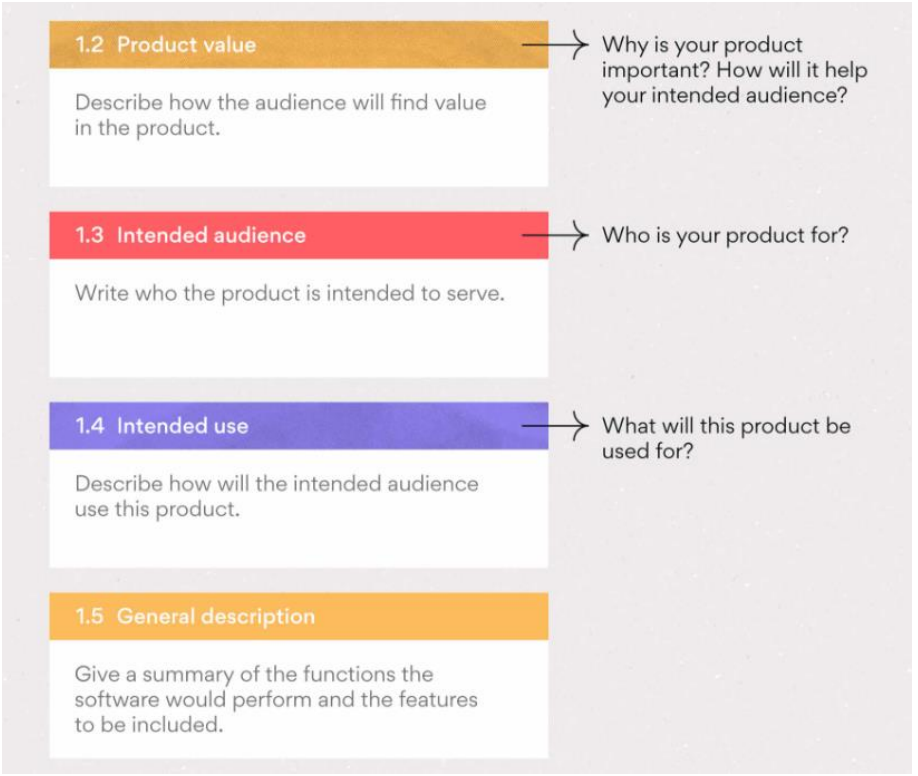
- Thu thập và phân tích yêu cầu:
  - **“Các vấn đề hiện tại là gì? “Chúng ta muốn gì? “”.**
- Thiết kế:
  - **“Làm thế nào chúng xây được những gì chúng ta muốn?”**
- Phát triển:
  - **“Tạo những gì chúng ta muốn”.**
- Kiểm thử:
  - **“Chúng ta đã đạt được những gì chúng ta muốn chưa? “**
- Triển khai:
  - **“Hãy bắt đầu sử dụng những gì chúng ta đã xây dựng”**
- Bảo trì/Duy trì:
  - **“Hãy giữ cho sản phẩm của chúng ta ổn định”**

Thu thập và phân tích yêu cầu

- Giai đoạn **Thu thập và phân tích và yêu cầu** bao gồm việc khám phá tình hình hiện tại, nhu cầu và ràng buộc của các bên liên quan, cơ sở hạ tầng hiện tại, v.v. và xác định vấn đề cần giải quyết bằng phần mềm.
- Sau khi thu thập các yêu cầu, nhóm phân tích kết quả để xác định những điều sau:
    - Có thể phát triển phần mềm theo những yêu cầu này không? Có thể được thực hiện với ngân sách không?
    - Có bất kỳ rủi ro nào đối với lịch trình phát triển không? Nếu có, chúng là gì?
    - Phần mềm sẽ được kiểm tra như thế nào?
    - Phần mềm sẽ được chuyển giao khi nào và như thế nào?
- **Đầu ra:** Vào cuối giai đoạn này, phương pháp **Waterfall** khuyến nghị tạo tài liệu "**Đặc tả yêu cầu phần mềm**" (Software Requirement Specification - SRS), trong đó nêu rõ các yêu cầu và phạm vi phần mềm, đồng thời xác nhận điều này một cách tỉ mỉ với các bên liên quan.

Tài liệu đặc tả phần mềm (SRS)

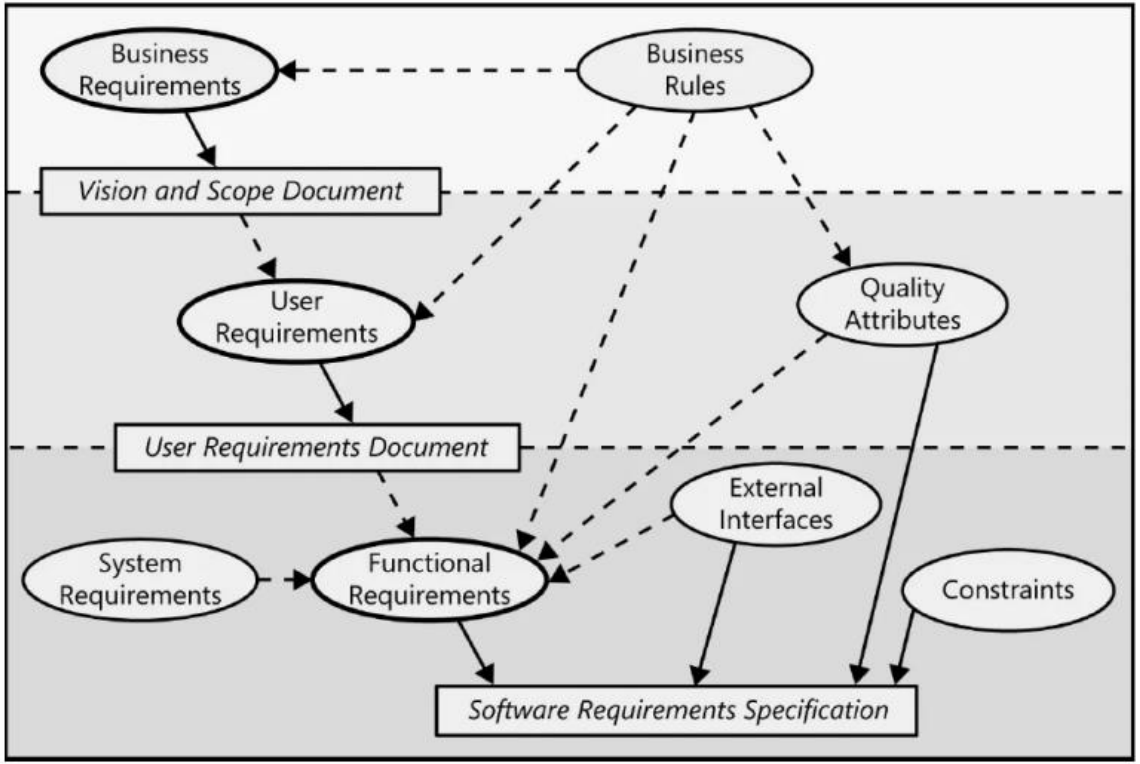




Tài liệu đặc tả phần mềm (SRS)

Table of Contents

Tài liệu đặc tả phần mềm (SRS)



Mô hình mối quan hệ giữa một số loại thông tin yêu cầu. Nguồn: Software Requirements by Karl Wiegers Joy Beatty

Thiết kế và hiện thực

Thiết kế (Design)

- Trong giai đoạn Thiết kế, các kiến trúc sư và nhà phát triển phần mềm thiết kế phần mềm dựa trên SRS được cung cấp.
- Ở cuối giai đoạn, đội ngũ thiết kế tạo ra các tài liệu thiết kế mức cao - High-Level Design (HLD); và thiết kế Mức Thấp - Low-Level Design (LLD).

Hiện thực (implementation)

- Giai đoạn hiện thực còn được gọi là giai đoạn lập trình phần mềm (coding)
- Vì tất cả các thành phần và mô-đun được xây dựng trong giai đoạn này, nên đó là giai đoạn dài nhất của vòng đời.
- Vào cuối giai đoạn, mã chức năng (functional code) thực hiện tất cả các yêu cầu của khách hàng đã sẵn sàng để được kiểm tra.

Kiểm thử, Triển khai, Bảo trì

Kiểm thử (Testing)

- Thực hiện kiểm thử mã nguồn đã phát triển trong môi trường kiểm thử.
- Bao gồm các dạng kiểm thử: Functional testing, integration testing, performance testing and security testing.
- Quá trình kiểm tra tiếp tục cho đến khi tất cả các mã đều không có lỗi và vượt qua tất cả các bài kiểm tra.
- Cuối giai đoạn này, một phần mềm chất lượng cao, không có lỗi, đã sẵn sàng để đưa ra sử dụng (production).

Triển khai (Deployment)

- Phần mềm được cài đặt trên môi trường sản phẩm sẽ chạy thực tế (production environment)
- Vào cuối giai đoạn, người quản lý sản phẩm phát hành phần mềm cuối cùng cho người dùng cuối.

Duy trì/Bảo trì (Maintenance)

- Trong suốt giai đoạn Bảo trì, đội ngũ sẽ:
  - Cung cấp sự hỗ trợ cho khách hàng
  - Gỡ lỗi được tìm thấy trên sản phẩm phần mềm
  - Cải tiến phần mềm
  - Thu thập thông tin mới từ khách hàng
- Cuối cùng, nhóm sẽ làm việc trên vòng lặp tiếp theo của SDLC và phiên bản tiếp theo của phần mềm.

SDLC

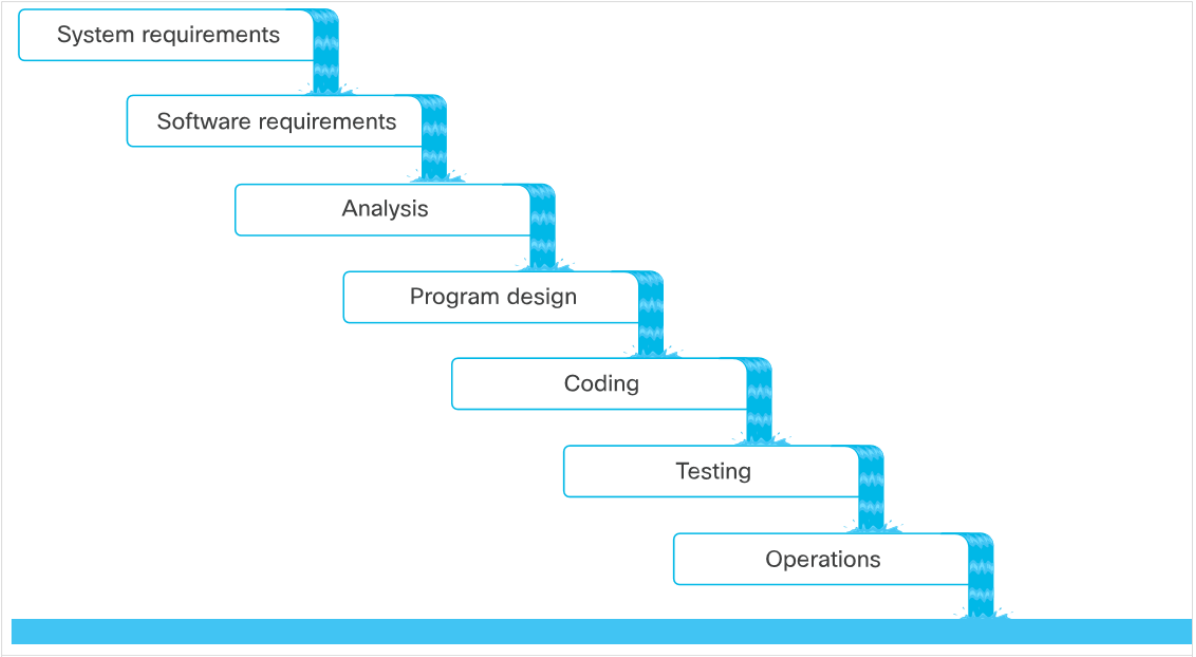
Giai đoạn	Đầu ra	Chịu trách nhiệm
REQUIREMENT/PLANNING Lấy yêu cầu/Lên kế hoạch	Tài liệu Yêu cầu KH **Tài liệu đề xuất giải pháp (Proposal)** **Tài liệu Quản lý dự án (Project Plan)**	Sales, BA, PM, Customer
ANALYSIS Phân tích	**Tài liệu Đặc tả yêu cầu**	PM, BA, Senior, Software Architect
DESIGN Thiết kế	**Hồ sơ thiết kế, Flowchart, Diagram** …	Designer, BA
CODING Lập trình	Sản phẩm (Source Code)	Developers
TESTING Kiểm thử	Test case, Test plan, Bug Report …	QA, Tester
DEPLOYMENT & MAINTENANCE Cài đặt và bảo trì	Tài liệu Đặc tả hệ thống Tài liệu Hướng dẫn sử dụng Tài liệu Cấu hình và cài đặt	DevOps

Phương pháp luận phát triển phần mềm

- Phương pháp luận phát triển phần mềm (software development methodology) thường được biết đến với thuật ngữ mô hình Chu kỳ Phát triển phần mềm (SDLC Model).
- Có 02 phương pháp phát triển phần mềm phổ biến:
  - Traditional SDLC: Waterfall
  - Agile SDLC: Scrum, Lean
- Việc sử dụng các phương pháp phát triển phần mềm, phụ thuộc vào:
  - Loại dự án (project)
  - Độ dài/thời gian thực hiện dự án (length of project)
  - Số lượng nhân lực của đội ngũ phát triển phần mềm (size of team).
  - Kinh phí

Phương pháp phát triển phần mềm Waterfall

- Mô hình Thác nước (Waterfall) - Winston W. Royce, 1970.
  - Bao gồm 7 giai đoạn



Phương pháp phát triển phần mềm Waterfall

- Mô hình Thác nước (Waterfall) - Winston W. Royce, 1970.
  - Các ưu điểm
    - Đơn giản, nổi tiếng, dễ hiểu, dễ sử dụng, dễ quản lý
    - Kế hoạch rõ ràng, các tài liệu được hoàn thành sau mỗi giai đoạn
    - Các yêu cầu: cung cấp sớm cho các người kiểm thử
    - Người quản lý dự án (PM): lập kế hoạch, kiểm soát chặt chẽ
  - Các nhược điểm
    - Không linh hoạt, khó thích ứng khi có thay đổi
    - Thời gian phân phối sản phẩm lâu hơn
    - Không phù hợp với các dự án kéo dài và tiếp diễn lâu
    - Nhiều nguy cơ (risk) và không chắc chắn (uncertainty)
    - Lãng phí nguồn lực

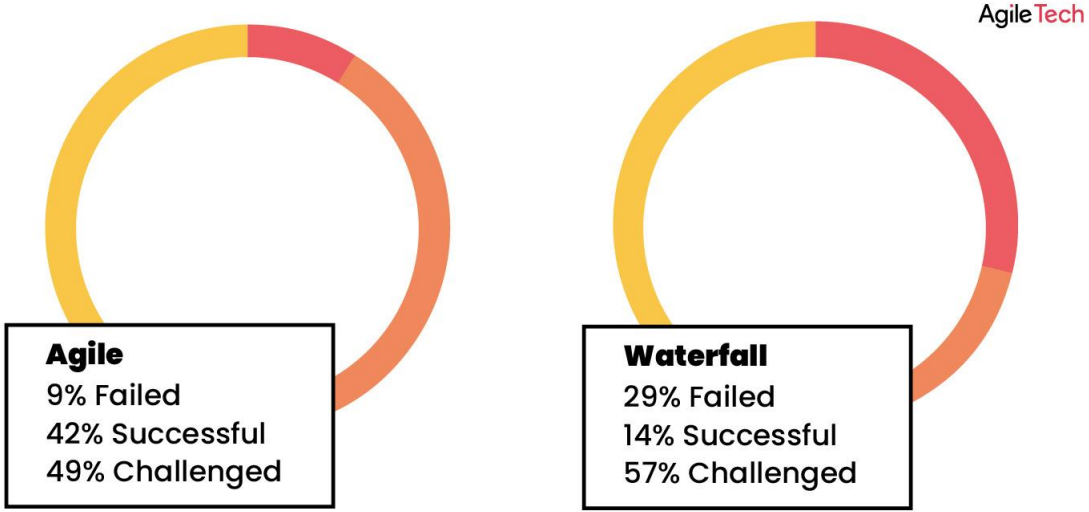
Dự án phù hợp: có các yêu cầu được xác định rõ ràng, đầy đủ và cố địnhhoặc ít thay đổi, nhân lực có kinh nghiệm, nắm vững công nghệ

Phương pháp phát triển phần mềm truyền thống

- Khi nào nên chọn mô hình phát triển phần mềm truyền thống (Plan-driven):
  - Các giai đoạn và yêu cầu được xác định rõ rang
  - Đưa ra sản phẩm mẫu chạy thử (protype) là một việc khó
  - Biết trước tất cả các yêu cầu trước khi bắt đầu dự án
  - Việc thay đổi không phải là ưu tiên
  - Các vai trò (role) được định nghĩa chat chẽ
  - Được khuyến nghị sử dụng trong các dự án phát triển với thời gian dài, chi phí cao
  - Tài liệu hóa chi tiết, toàn bộ quá trình phát triển
- Một số mô hình phát triển phần mềm truyền thống (Planning-driven)
  - Waterfall
  - Iterative Model
  - Rational Unified Process (RUP)
  - Spiral Model

Phương pháp luận phát triển phần mềm

- Thống kê sự thành công của dự án phần mềm dựa trên 02 phương pháp.



*Source: The CHAOS report from the Standish Group (2015)

Phương pháp phát triển phần mềm Agile

- Phương pháp Agile là phương pháp phát triển phần mềm tập trung vào tính linh động và hướng khách hàng (customer-focused).
- Theo Agile Manifesto, giá trị của phương pháp Agile bao gồm:
  - Chú trọng vào con người và tương tác của các thành viên hơn là qui trình vàcông cụ hỗ trợ
  - Chú trọng vào phần mềm hoạt động như yêu cầu hơn là tài liệu thiết kế
  - Chú trọng việc cộng tác với khách hàng hơn là các hợp đồng thoả thuận
  - Chú trọng khả năng thích ứng với các sự thay đổi hơn là tuân theo kế hoạch
- Agile manifesto bao gồm 12 nguyên tắc khác nhau:

Agile Manifesto Principles			
Tập trung vào khách hàng	Hợp tác giữa các bên	Ưu tiên phần mềm hoạt động	Đơn giản
Chấp nhận sự thay đổi và thích ứng	Đội ngũ có động lực	Làm việc với tốc độ bền vững	Các nhóm tự tổ chức
Thường xuyên phân phối các phiên bản phần mềm hoạt động	Trao đổi trực tiếp	Môi trường linh hoạt, tốc độ (Agile)	Liên tục cải thiện

Các phương pháp phát triển phần mềm Agile

- Các phương pháp Agile phổ biến:
  - **Scrum:** Scrum tập trung vào các nhóm nhỏ, tự tổ chức, nhóm họp hàng ngày trong thời gian ngắn và làm việc trong các cuộc chạy nước rút (sprint) lặp đi lặp lại.
  - **Lean:** Phương pháp Lean nhấn mạnh vào việc loại bỏ nỗ lực hao phí trong việc lập kế hoạch và thực hiện, đồng thời giảm tải nhận thức của lập trình viên.
  - **Extreme Programming (XP):** XP hướng tới giải quyết các loại vấn đề cụ thể về chất lượng mà các nhóm phát triển phần mềm phải đối mặt.
  - **Feature-Driven Development (FDD):** FDD quy định rằng việc phát triển phần mềm phải được tiến hành theo mô hình tổng thể, được chia nhỏ, lập kế hoạch, thiết kế và xây dựng theo từng tính năng.

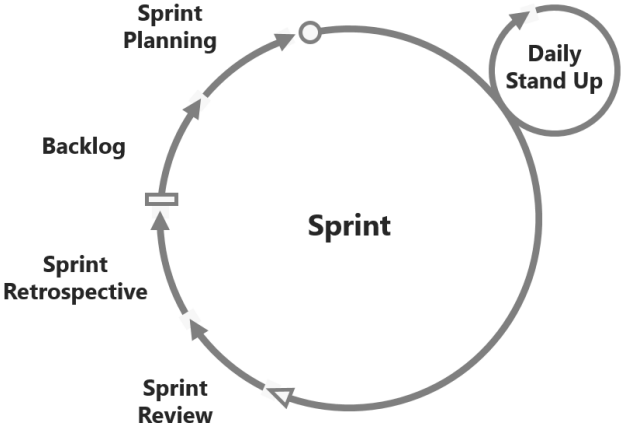
Scrum

- **Sprints**
  - Sprint là một khoảng thời gian cụ thể, thường từ 2-4 tuần, trong đó, mỗi nhóm đảm nhận nhiều nhiệm vụ (còn được gọi là User Story - câu chuyện của người dùng) mà họ cảm thấy có thể hoàn thành. Khi sprint kết thúc, phần mềm sẽ hoạt động và có thể phân phối được.
  - Khoảng thời gian của sprint được xác định trước khi quá trình bắt đầu và hiếm khi thay đổi.
- **Backlog**
  - Công việc tồn đọng (BackLog) bao gồm tất cả các tính năng của phần mềm, nằm trong danh sách ưu tiên.
- **User stories**
  - Câu chuyện người dùng (user story) là một tuyên bố đơn giản về những gì người dùng (hoặc một vai trò) cần và tại sao. Mỗi câu chuyện của người dùng phải đủ nhỏ để một nhóm có thể hoàn thành nó trong một lần chạy nước rút (sprint).
  - Mẫu gợi ý của 01 user story như sau:

As a <user|role>, I would like to <action>, so that <value|benefit>

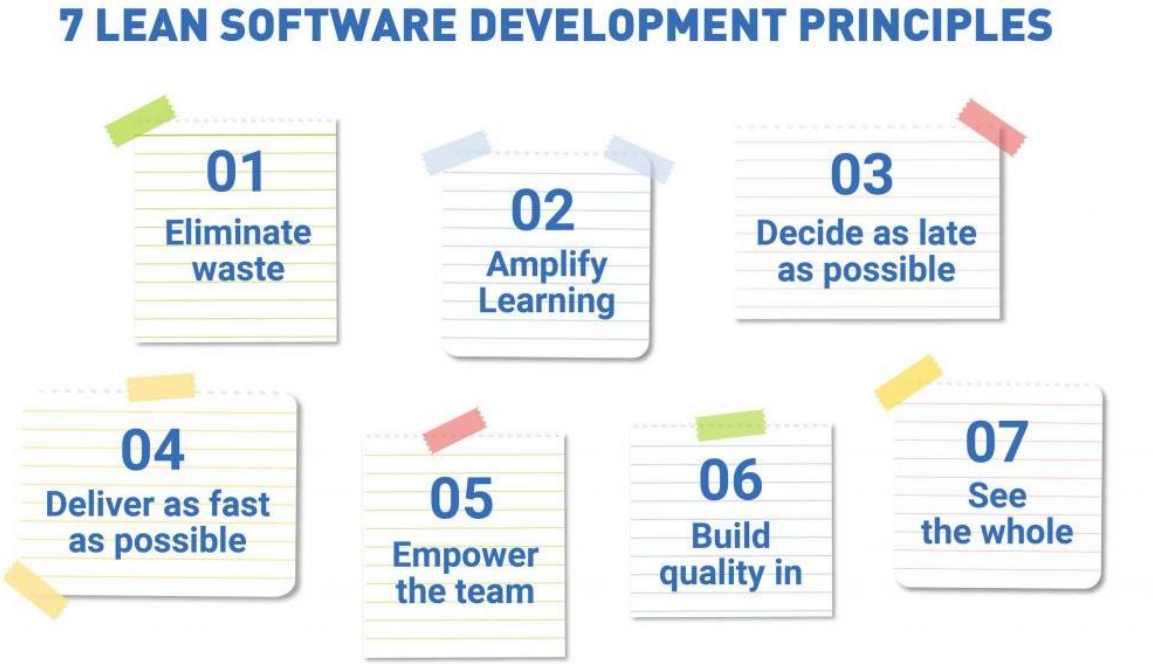
Nhóm Scrum (Scrum Teams)

- Nhóm Scrum có khả năng làm việc phối hợp (cross-functional), hợp tác, tự quản lý và tự trao quyền.
- Các nhóm scrum không được lớn hơn 10 cá nhân. Một Scrum Team điển hình thì nên trong khoảng từ 3-9 người, bao gồm cả ***Product Owner*** và ***Scrum Master***.
  - Product Owner – “What” Guy
  - Scrum Master – “How” Guy
- Scrum master nên có một cuộc họp trực tiếp hàng ngày với nhóm vào một thời gian cố định hàng ngày, không quá 15 phút.
- Mục đích là để "điểm lại" các nhiệm vụ (task) quan trọng đã hoàn thành, đang thực hiện hoặc sắp bắt đầu.



Phương pháp phát triển phần mềm Lean

- Phát triển phần mềm Lean dựa trên nguyên tắc **Lean Manufacturing**, tập trung vào việc giảm thiểu hao phí (waste) và tối đa hoá giá trị đối với khách hàng.



Phương pháp phát triển phần mềm Lean

- Qui tắc Lean bao gồm 7 bước, được nêu trong sách “Lean Software Development: An Agile Toolkit,” cụ thể như sau:
  - Giảm thiểu hao phí (Eliminate waste): xác định và triệt tiêu hao phí trong qui trình SDLC
  - Tăng cường việc học tập (Amplify learning): thông qua việc thực hành; thông qua sự tương tác và tích lũy kinh nghiệm từ những gì khách hàng muốn/thích.
  - Quyết định càng muộn càng tốt (Decide as late as possible)
  - Phân phối càng nhanh càng tốt (Deliver as fast as possible)
  - Trao quyền cho nhóm (Empower the team): Tạo sự hợp tác làm việc phối hợp từ bên trong nhóm hơn là từ người quản lý cấp trên
  - Xây dựng sự toàn vẹn từ bên trong (Build integrity in)
  - Tối ưu hóa toàn bộ (Optimize the whole): suy nghĩ cho thời gian dài hạn, ưu tiên Khách hàng > Công ty > Nhóm làm việc > Cá nhân

Phương pháp phát triển phần mềm Lean

Giảm thiểu hao phí (waste)

- Đây là nguyên tắc nền tảng cơ bản nhất của Lean.
- Có 07 dạng hao phí khác nhau trong phát triển phần mềm:
  - Hoàn thành công việc một phần (Partially done work)
  - Các quy trình gia tăng (Extra processes)
  - Các chức năng gia tăng (Extra features)
  - Hoán đổi nhiệm vụ công việc (Task switching)
  - Chờ đợi (Waiting)
  - Di chuyển (Motion)
  - Khuyết điểm (Defects)

Phương pháp phát triển phần mềm Lean

Tăng cường sự học hỏi với những sprint ngắn (Amplify Learning with Short Sprints)

- Để tinh chỉnh phần mềm, cần có nhiều vòng lặp ngắn hạn thường xuyên của phần mềm, cho phép:
  - Lập trình viên học hỏi nhanh hơn.
  - Khách hàng có thể phản hồi sớm hơn.
  - Các chức năng có thể được tùy chỉnh để mang đến nhiều lợi ích hơn cho khách hàng.

Đưa ra quyết định càng muộn càng tốt (Decide as Late as Possible)

- Khi chưa chắc chắn, tốt nhất là nên delay việc đưa ra quyết định càng muộn càng tốt. Vì đưa ra quyết định dựa trên thực tế vẫn tốt hơn dựa trên ý kiến hoặc suy đoán.

Phân phối sản phẩm càng nhanh càng tốt (Deliver as Fast as Possible)

Deliver As Fast as Possible	
Enables customers to provide feedback	Doesn't allow customers to change their mind
Enables developers to amplify learning	Makes everyone take decisions faster
Provides customers the required features	Produces less waste

Phương pháp phát triển phần mềm Lean

Trao quyền cho nhóm (Empower the Team)

- Mỗi người cần được cho phép đưa ra các quyết định trong lĩnh vực chuyênmôn của họ.

Xây dựng sự toàn vẹn từ bên trong (Build Integrity In)

- Sự toàn vẹn của phần mềm là khi nó đáp ứng được yêu cầu cũng như hữuích cho khách hàng.

Tối ưu toàn bộ (Optimize the Whole)

- Phần mềm cần được phát triển một cách toàn diện, liên kết. Giá trị của phầnmềm sẽ bị ảnh hưởng đến như mỗi chuyên gia chỉ tập trung vào chuyên môncủa mình mà không xem xét quyết định của họ có ảnh hưởng đến các phầncòn lại của phần mềm hay không.

Phương pháp phát triển phần mềm Lean

Ưu điểm

- Phù hợp với môi trường năng động, hỗn loạn
- Hỗ trợ các yêu cầu phát sinh, thay đổi nhanh chóng
- Tránh chi phí tài liệu
- Trao quyền/khuyến khích mọi người

Nhược điểm

- Cần người có năng lực cao, có tinh thần tự giác
- Khó khăn trong các hệ thống quan trọng về an toàn
- Vấn đề ở các dự án lớn

Truyền thống vs Agile

Đặc điểm	Truyền thống (Waterfall)	Agile (Scrum)
Mục tiêu	Tăng sự thành công của dự án qua việc lặp lại những kết quả tốt của các dự án đã thành công	Đáp ứng sự thay đổi của khách hàng mộtcách tốt nhất, hiệu chỉnh phù hợp với từngkhách hàng và dự án, chuyển giao nhanh
Quy trình	Chặt chẽ, các bước được định nghĩa rõ ràng	Quy trình không chặt chẽ, được hiểu ngầm
Tài liệu	Tài liệu, biểu mẫu nhiều, rõ ràng	Ít viết tài liệu, dựa vào hiểu ngầm
Kế hoạch	Kế hoạch chi tiết, áp dụng xuyên suốt dự án	Có kế hoạch nhưng ở mức độ cao, không chi tiết
Quản lý	Dựa vào kế hoạch, so sánh kế hoạch với thực tế, chỉ tương tác với khách hàng khi cần thiết	Trông đợi vào sự hợp tác giữa các thành viên trong nhóm, tăng sự chủ động của thànhviên, đánh giá tiến độ dựa trên khả năng đáp ứng yêu cầu của khách hàng, tương tác với khách hàng thường xuyên
Các dự án phù hợp	Lớn, nhiều người Yêu cầu về độ an toàn cao	Nhỏ, ít người (<20 người) Dự án có thay đổi nhiều và nhanh Dự án ngắn

Thảo luận

Phương pháp Agile có phù hợp với những dự án có yêu cầu về mức độ an toàn và chính xác cao (critical systems) hay không? Giải thích lí do.

Tích hợp AI vào trong phát triển phần mềm

Liệu rằng có thể tích hợp AI vào trong SDLC?

Tích hợp AI vào trong phát triển phần mềm

1. Thu thập & phân tích yêu cầu

"Các vấn đề hiện tại là gì? Chúng ta muốn gì?"

- Thay vì **nhóm phát triển phân tích** kết quả để **xác định** các **thông tin** sau:
  - Có thể phát triển phần mềm đáp ứng các yêu cầu này được hay không? Cóđảm bảo ngân sách hay không?
  - Có rủi ro nào trong kế hoạch phát triển không? Nếu có thì là gì?
  - Phần mềm sẽ được kiểm thử như thế nào?
  - Phần mềm cần được chuyển giao khi nào và bằng cách nào?

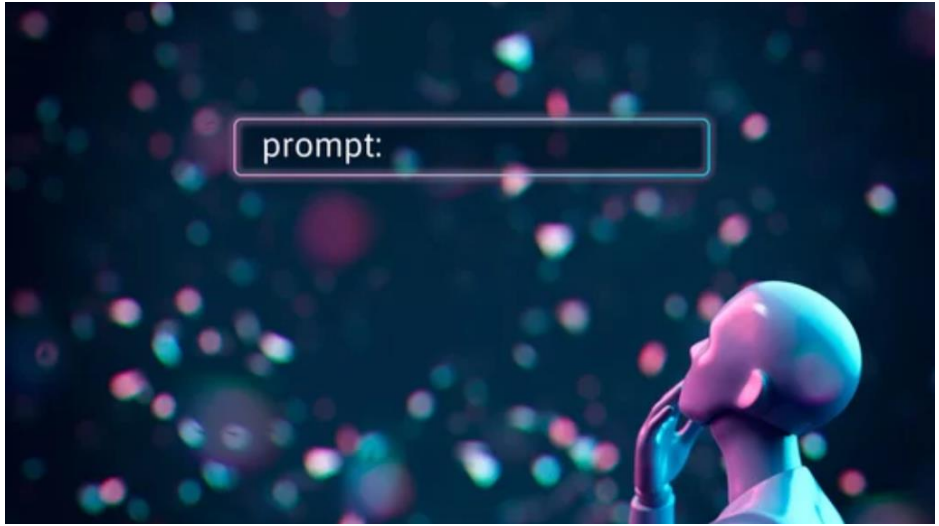
=> Có thể sử dụng các trợ lí AI như **ChatGPT, Gemini**, … để giúp mình trả lời cho các câu hỏi trên. Bên cạnh đó, cũng có thể sử dụng các hệ thống AI giúp xây dựng phần mềm ứng dụng tự động như **PartyRock**.

Tích hợp AI vào trong phát triển phần mềm

2. Thiết kế

“Làm thế nào chúng ta xây được những gì chúng ta muốn?”

- Thay vì **nhóm phát triển tự thiết kế dựa trên SRS** => Có thể sử dụng **LLM-based Architecture Advisor** để gợi ý design patterns, kiến trúc microservices.



Tích hợp AI vào trong phát triển phần mềm

3. Phát triển (Coding)

"Tạo những gì chúng ta muốn."

- **Code Generation Models**:
  - *Copilot*
  - *Amazon CodeWhisperer*
  - *Tabnine*
  - …
- **Static Analysis AI**: DeepCode, CodeQL, …

Tích hợp AI vào trong phát triển phần mềm

4. Kiểm thử

“Chúng ta đã đạt được những gì chúng ta muốn chưa?”

- **Test Case Generation Models**:
  - LLM (GPT-4, Codex) sinh unit test, integration test.
  - EvoSuite
  - ...
- **AI fuzzing models**

Tích hợp AI vào trong phát triển phần mềm

5. Triển khai

“Hãy bắt đầu sử dụng những gì chúng ta đã xây dựng.”

- AI có thể hỗ trợ viết mã để quản lý cơ sở hạ tầng, tự động hóa việc tạo và cấu hình máy chủ, mạng và các tài nguyên khác. Dựa trên việc kết hợp các nền tảng như Docker, cơ sở hạ tầng dưới dạng mã với AWS CloudFormation, Mô hình Ứng dụng Không Máy chủ AWS (SAM) hoặc HashiCorp Terraform.

Tích hợp AI vào trong phát triển phần mềm

6. Bảo trì / Duy trì

"Hãy giữ cho sản phẩm của chúng ta ổn định."

- Sau khi triển khai, phần mềm bước vào giai đoạn bảo trì, nơi nó liên tục được theo dõi, cập nhật và cải tiến dựa trên phản hồi của người dùng, các yêu cầu thay đổi và bối cảnh công nghệ đang phát triển. AI có thể giúp xử lý và phân tích dữ liệu đầu ra khổng lồ của các công cụ quan sát, hỗ trợ khắc phục sự cố hiệu quả và phát hiện vấn đề chủ động.

⇒ Tuy nhiên, cho đến thời điểm hiện tại vẫn chưa có một hệ thống AI hoàn chỉnh nào có thể giúp nhà phát triển hoàn thiện hết toàn bộ các bước trong quy trình phát triển phần mềm.

Bài tập

Giả sử bạn đang là trưởng dự án (project manager) cho một công ty **chuyên về phát triển và cung cấp các dịch vụ phần mềm** hỗ trợ nông dân trong việc quản lý sản xuất và phân phối sản phẩm nông nghiệp. Công ty có tên *Bạn Nhà Nông* hay *BNN*. Khách hàng tiềm năng của công ty là công ty trách nhiệm hữu hạn *Hai Lúa*. Công ty Hai Lúa có kế hoạch phát triển một hệ thống thương mại điện tử *hailua.com.vn* nhằm cung cấp các dịch vụ cho việc quảng bá, mua và bán các sản phẩm nông nghiệp ở Việt Nam. Mô hình hoạt động giống như ebay.com. Giám đốc công ty BNN phân công bạn phụ trách dự án này.

Dự án tương đối lớn với số thành viên khoảng 15-20 người và thời gian thực hiện từ 8 tháng đến 1 năm.

- **a.** Hãy nêu ra các điểm thuận lợi và bất lợi nếu áp dụng phương pháp Waterfall, Rational Unified Process (RUP), Lean, và Scrum.
- **b.** Hãy đưa ra các giả định (hay điều kiện) của dự án để mỗi phương pháp trên là phù hợp nhất với dự án.